(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開平9－245015
(43)公開日　平成９年(1997)９月19日

(51)Int.Cl.[6]	識別記号	庁内整理番号	ＦＩ		技術表示箇所
Ｇ０６Ｆ　17/00			Ｇ０６Ｆ　15/20	Ｚ	

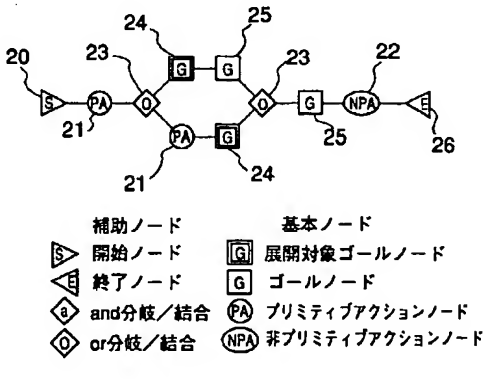
審査請求　未請求　請求項の数10　ＯＬ　（全 14 頁）

(21)出願番号　特願平8－56402
(22)出願日　平成８年(1996)３月13日

(71)出願人　000003078
株式会社東芝
神奈川県川崎市幸区堀川町72番地
(72)発明者　平原　明
東京都府中市東芝町１番地　株式会社東芝府中工場内
(74)代理人　弁理士　鈴江　武彦

(54)【発明の名称】　階層的プランニング方法

(57)【要約】
【課題】　本発明は、無駄なアクションを含むプランの生成を抑制し、同時に全体のプラン生成効率を向上させる階層的プランニング方法を提供する。
【解決手段】　開始位置における初期状態に対して基本的なアクションを順次実行にすることにより、初期状態から目標状態へ状態を変化させるアクションの系列を計画として生成する方法であって、開始位置から順次接続されたゴール等をノードとする手続きネットワークを基本的なアクションからなる系列に展開することで計画を得る階層的プランニング方法において、手続きネットワーク上のゴール等を展開するとき、その展開対象のノードとしては、開始位置から基本的なアクションのノード及び又は補助的なノードのみを介して接続されるゴール等のノードを選択する階層的プランニング方法である。



1

【特許請求の範囲】

【請求項１】　開始位置における初期状態に対して基本的なアクションを順次実行にすることにより、前記初期状態から目標状態へ状態を変化させるアクションの系列を計画として生成する方法であって、前記開始位置から順次接続されたアクションやゴールをノードとする手続きネットワークを前記基本的なアクションからなる系列に展開することで前記計画を得る階層的プランニング方法において、前記手続きネットワーク上の前記アクションやゴールを展開するとき、その展開対象のノードとしては、前記開始位置から前記基本的なアクションのノード及び又は補助的なノードのみを介して接続される前記アクションやゴールのノードを選択することを特徴とする階層的プランニング方法。

【請求項２】　選択され得る展開対象のノードが複数ある場合、予め定められた戦略に基づき、実際に展開される展開対象を決定することを特徴とする請求項１記載の階層的プランニング方法。

【請求項３】　前記予め定められた戦略は、前記開始位置から介される前記基本的なアクションのノード数が最も少ない前記選択され得る展開対象を、前記実際に展開される展開対象として決定する戦略であることを特徴とする請求項２載の階層的プランニング方法。

【請求項４】　前記予め定められた戦略は、前記開始位置から介される前記基本的なアクションのノード数が最も多い前記選択され得る展開対象を、前記実際に展開される展開対象として決定する戦略であることを特徴とする請求項２記載の階層的プランニング方法。

【請求項５】　前記アクションやゴールが展開されることにより関連するノードでの状態が変化したときに、そのノードにおける状態の変化分を当該ノードでの状態に代えて保存することを特徴とする請求項１乃至４のうち何れか１項記載の階層的プランニング方法。

【請求項６】　前記手続きネットワーク上におけるアクションやゴールのノードを展開する場合に、その展開に適用する展開知識が欠如しているとき、予め与えられた基本的なアクションの情報を用い、試行錯誤による探索で当該アクションやゴールを達成する基本的なアクションの系列を求めることを特徴とする請求項１項記載の階層的プランニング方法。

【請求項７】　前記探索で求められた基本的なアクションの系列，その探索開始時の状態及び対応するアクションやゴールの情報を保存し、前記アクションやゴールと同様なノードについて探索開始時と同じ状態なったとき、再び探索することなく前記探索で求められた基本的なアクションの系列を展開することを特徴とする請求項６項記載の階層的プランニング方法。

【請求項８】　開始位置における初期状態に対して基本的なアクションを順次実行にすることにより、前記初期状態から目標状態へ状態を変化させるアクションの系列

2

を計画として生成する方法であって、前記開始位置から順次接続されたアクションやゴールをノードとする手続きネットワークを、前記基本的なアクションからなる系列に展開知識を用いて展開することで前記計画を得る階層的プランニング方法において、前記手続きネットワーク上の前記アクションやゴールに対する展開知識におけるアクションやゴールの状態を、複数の状態述語によって表現することを特徴とする階層的プランニング方法。

【請求項９】　前記手続きネットワーク上の前記アクションやゴールを展開するとき、その展開対象のノードとしては、前記開始位置から前記基本的なアクションのノード及び又は補助的なノードのみを介して接続される前記アクションやゴールのノードを選択することを特徴とする請求項８記載の階層的プランニング方法。

【請求項１０】　前記手続きネットワーク上のノードが、状態述語によって定義された禁止状態に陥っているか否かを監視することを特徴とする請求項１又は８記載の階層的プランニング方法。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】この発明は、種々の分野で発生する行動計画を決定するためのプラン（計画）を作成する階層的プランニング方法に関するものである。

【０００２】

【従来の技術】近年、種々の分野で発生する事象に対し、その行動計画を決定するためのプラン作成について、計算機による階層型プランナを用いることが行われるようになっている。

【０００３】簡単な例では、計算機ネットワークの電源をおとすべき順序を決めるなどがある。また、例えば電力系統がダウンした後に、これを復旧するときの機器操作の順番を決めるプランを作成するためのエキスパートシステムとして、この階層型プランナを用いることが考えられている。

【０００４】ここで、プランとは、初期状態から目標状態へ状態を変化させるアクションの系列のことをいい、プランニングとは、初期状態、目標状態、アクションの３つが与えられた時に、実行可能なプランを求めることをいう。

【０００５】このプランニングの従来方法の１つとして、手続きネットワークを用いた階層的なプランニング方法がある。手続きネットワークは、プランニング過程のプランの内部表現形式で、ゴールノードとアクションノードの２種類の基本ノードとプランの開始、終了などを示す補助ノードからなる。アクションノードには基本的なアクションを表すプリミティブアクションノードと抽象的なアクションを表す非プリミティブアクションノードの２つがある。

【０００６】目標状態は手続きネットワーク上の非プリミティブなアクションノードやゴールノードとして表さ

れる。プランニングは、ゴールノードやアクションノードを、より具体的なゴール（サブゴール）ノードやアクションノードからなるサブネットワーク置き換えることによって進む。この置き換えをノードの展開と言う。展開はあらかじめ与えられた知識に従って行う。全ての基本ノードがプリミティブアクションになった時が、プランの候補が得られた時である。

【０００７】最後にアクション間の相互作用を検出し、アクション間に適切な順序づけをして相互作用を解消する。解消が不可能ならば再プランニングする。ここで、相互作用とは、手続きネットワーク上で並列に実行可能となっているプリミティブアクション間で、一方のプリミティブアクションの効果が他方のプリミティブアクションの前提条件を打ち消す場合をいう。相互作用を全て解消できたら、それを最終的なプランとして出力する。

【０００８】この階層型プランナによるプラン作成の簡単な例を図９に示す。図９は階層型プランナを用いてプラン候補を作成する例を示す図である。同図（ａ'）においてスタート時には、まず、手続きネットワークは、開始ノード１０１と、ゴールノード１０２と、終了ノード１０３とのみからなっている。開始ノード１０１には、初期状態が与えられ、ゴールノード１０２により目標状態が表されている。また、初期状態がゴールノード１０２の内容により処理されると目標状態が達成されて終了ノード１０３に至る。

【０００９】プランを作成するには、まず、このゴールノード１０２をあらかじめ与えられたゴール展開知識により展開する。ゴールノード１０２を展開し、具体的なゴールノード１０５，１０６，１０８にすることにより手続きネットワークは、図９（ｂ）に示される状態となる。

【００１０】このゴールノード１０５，１０６，１０８夫々をゴール展開知識によりさらに展開すると、図９（ｃ）に示す状態となる。同図（ｃ）においては、すべてのゴールノードが展開され、基本ノードはすべてプリミティブアクションノード１０９，１１０，１１１，１１３，１１４，１１６となっている。なお、これらのプリミティブアクションノードを接続するために、ａｎｄ分岐／結合１０４，１０７、ｏｒ分岐／結合１１２，１１５が用いられている。

【００１１】この完成したプランにおいて、開始ノード１０１の初期状態を各ノードで順次処理し、終了ノード１０３に至ると最終的な目的状態が得られることとなる。なお、上記例では、展開されるべきノードとして、ゴールノードのみを示したが、非プリミティブアクションのノード（非プリミティブアクションノード，アクションノード）もゴールノードと同様に展開知識（アクション展開知識）によって、プリミティブアクションノード等に展開されるものであり、非プリミティブアクションノードは、ゴールノードの場合よりもより抽象的な内容をアクションとして取り扱うときに用いられる。

【００１２】また、完成したプラン候補もしくは展開途中段階の適宜な状態で、相互作用をチェックする。図１０の例で示すように、相互作用は、プリミティブアクションノードがａｎｄ分岐／結合で並列に配置されたときに生じる可能性がある。

【００１３】図１０は階層型プランナにおける相互作用のチェックを説明するための図である。同図の例では、相互作用は、プリミティブアクションノード１２３<－>プリミティブアクションノード１２８、プリミティブアクションノード１２３<－>プリミティブアクションノード１２９、プリミティブアクションノード１２５<－>プリミティブアクションノード１２６、プリミティブアクションノード１２５<－>プリミティブアクションノード１２８、プリミティブアクションノード１２５<－>プリミティブアクションノード１２９、プリミティブアクションノード１２６<－>プリミティブアクションノード１２８、プリミティブアクションノード１２６<－>プリミティブアクションノード１２９間においてチェックされる。

【００１４】つまり、同時に実行し得るプリミティブアクションのすべての対に対して行われることとなる。ここで、いくつかの対で相互作用が検出されて、順序付けを行う必要が生じた時には、すべての相互作用を解消するように順序付けが行われる。

【００１５】ところで、上記した図９は、ごく簡単な例を示しているが、実際の階層型プランナは極めて複雑なプランを作成可能とすることを目的としている。このような考え方から、各ゴールノードやアクションノードは引数として変数を有するままで展開が行われていく。つまり、そのプランナ装置の有するルールに従って、ある段階における全ゴールノード又はアクションノードの中から適宜なゴールノード等が選択され展開されるのである。

【００１６】したがって、上記ゴールノード等を展開していく過程で不適切な展開があった場合には、不適切な展開をキャンセルして展開をやり直す等の探索を行いつつプランニングが行われる。

【００１７】このように変数を有するまま、その段階でのゴールノード中の適宜なゴールノードを展開していくというやり方は、高い自由度をもって探索を行うことができ、また、少ない探索量でプランニングを実現できる可能性を有している。

【００１８】

【発明が解決しようとする課題】しかしながら、従来の変数を有するまま適宜なゴールを展開する方法においては、展開の自由度が高いゆえに逆に、階層的プランナ制作者がゴールノードを展開するための展開知識を作成するのが難しいという問題点がある。

【００１９】また、この方法では、上記したように少な

い探索量でプランニングを実現できる場合もあるが、一方で、展開知識の条件に合うノードはすべて展開可能ノードとしているために、無駄なアクションを含むプランを生成する場合がある。

【００２０】簡単なプランの場合はこのようなことはあまり考えられないが、複雑なプランの場合、例えば、ある条件Ａが成立しないとアクションＸ，Ｙを実施できないのに、条件Ａを成立させるアクションＺのノード作成の前に、アクションＸ，Ｙを実施するノードを作成することがある。

【００２１】このような場合には条件Ａのアクションのノード作成時に、アクション逆Ｙ，逆Ｘと逆戻りさせるノードを作成した上で、アクションＺのノードを作成することになる。したがって、本来はアクションＺ，Ｘ，Ｙですむところを、この場合には、アクションＸ，Ｙ，逆Ｙ，逆Ｘ，Ｚ，Ｘ，Ｙとしなければならない。

【００２２】つまり、従来の方法は、少ない探索量で効率的なプランを作成できる可能性もあるが、同時に無駄なアクションを含むプランを作成する可能性もある。したがって、種々の分野で発生するいかなる行動計画に対しても、無駄の少ないプランを確実に作成するのは困難である。これは、プランの生成効率低下の原因ともなっている。

【００２３】また、従来の方法では、展開知識が完全でない場合、すなわち達成したいゴールを展開する知識がない場合には、プランの生成は失敗していた。一方、プランの対象世界はオブジェクトにより構成されるが、１つのノードが行うアクションは１つオブジェクトに関するものである。したがって、オブジェクトに対するアクションの結果得られる状態も基本ノードにおいて１つだけが記述される。しかしながら、従来の状態表現方法では、目標状態をサブゴールやアクションに展開する知識の記述が困難な場合があった。

【００２４】また、従来の方法では、制約条件として禁止された状態を効果的に除外されていなかった。本発明は、このような実情を考慮してなされたもので、その第１の目的は、無駄なアクションを含むプランの生成を抑制し、同時に全体のプラン生成効率を向上させる階層的プランニング方法を提供することにある。

【００２５】また、第２の目的は、知識が完全ではなく抜けがある場合にもプランニングを続行できるようにする階層的プランニング方法を提供することにある。さらに、第３の目的は、展開知識の記述しづらい問題にも対応することができる階層的プランニング方法を提供することにある。一方、第４の目的は、制約条件として禁止された状態を効果的に除外することを可能とする階層的プランニング方法を提供することにある。

【００２６】

【課題を解決するための手段】上記課題を解決するために、請求項１に対応する発明は、開始位置における初期

状態に対して基本的なアクションを順次実行にすることにより、初期状態から目標状態へ状態を変化させるアクションの系列を計画として生成する方法であって、開始位置から順次接続されたアクションやゴールをノードとする手続きネットワークを基本的なアクションからなる系列に展開することで計画を得る階層的プランニング方法において、手続きネットワーク上のアクションやゴールを展開するとき、その展開対象のノードとしては、開始位置から基本的なアクションのノード及び又は補助的なノードのみを介して接続されるアクションやゴールのノードを選択する階層的プランニング方法である。

【００２７】また、請求項２に対応する発明は、請求項１に対応する発明において、選択され得る展開対象のノードが複数ある場合、予め定められた戦略に基づき、実際に展開される展開対象を決定する階層的プランニング方法である。

【００２８】さらに、請求項３に対応する発明は、請求項２に対応する発明において、予め定められた戦略は、開始位置から介される基本的なアクションのノード数が最も少ない選択され得る展開対象を、実際に展開される展開対象として決定する戦略である階層的プランニング方法である。

【００２９】さらにまた、請求項４に対応する発明は、請求項２に対応する発明において、予め定められた戦略は、開始位置から介される基本的なアクションのノード数が最も多い前記選択され得る展開対象を、実際に展開される展開対象として決定する戦略である階層的プランニング方法である。

【００３０】一方、請求項５に対応する発明は、請求項１～４に対応する発明において、アクションやゴールが展開されることにより関連するノードでの状態が変化したときに、そのノードにおける状態の変化分を当該ノードでの状態に代えて保存する階層的プランニング方法である。

【００３１】また、請求項６に対応する発明は、請求項１に対応する発明において、手続きネットワーク上におけるアクションやゴールのノードを展開する場合に、その展開に適用する展開知識が欠如しているとき、予め与えられた基本的なアクションの情報を用い、試行錯誤による探索で当該アクションやゴールを達成する基本的なアクションの系列を求める階層的プランニング方法である。

【００３２】さらに、請求項７に対応する発明は、請求項６に対応する発明において、探索で求められた基本的なアクションの系列，その探索開始時の状態及び対応するアクションやゴールの情報を保存し、アクションやゴールと同様なノードについて探索開始時と同じ状態なったとき、再び探索することなく探索で求められた基本的なアクションの系列を展開する階層的プランニング方法である。

【００３３】さらにまた、請求項８に対応する発明は、開始位置における初期状態に対して基本的なアクションを順次実行にすることにより、初期状態から目標状態へ状態を変化させるアクションの系列を計画として生成する方法であって、開始位置から順次接続されたアクションやゴールをノードとする手続きネットワークを、基本的なアクションからなる系列に展開知識を用いて展開することで計画を得る階層的プランニング方法において、手続きネットワーク上のアクションやゴールに対する展開知識におけるアクションやゴールの状態を、複数の状態述語によって表現する階層的プランニング方法である。

【００３４】一方、請求項９に対応する発明は、請求項８に対応する発明において、手続きネットワーク上のアクションやゴールを展開するとき、その展開対象のノードとしては、開始位置から基本的なアクションのノード及び又は補助的なノードのみを介して接続されるアクションやゴールのノードを選択する階層的プランニング方法である。

【００３５】また、請求項１０に対応する発明は、請求項１又は８に対応する発明において、手続きネットワーク上のノードが、状態述語によって定義された禁止状態に陥っているか否かを監視する階層的プランニング方法である。

（作用）したがって、まず、請求項１に対応する発明の階層的プランニング方法においては、手続きネットワーク上のアクションやゴールを展開するとき、その展開対象のノードとしては、開始位置から基本的なアクションのノード及び又は補助的なノードのみを介して接続されるアクションやゴールのノードが選択される。

【００３６】このように、開始位置から最も近いゴールやアクションが順に展開されることで、その状態が確定しているものについて順に展開されることになる。したがって、その状態が未確定な引数や変数のままゴールやアクションが展開されることがない。

【００３７】したがって、蓋然性を高くしつつ状態が変化するような展開を確実に実施できるので、無駄なアクションを含むプランの生成を抑制し、同時に全体のプラン生成効率を向上させることができる。

【００３８】また、請求項２に対応する発明の階層的プランニング方法においては、請求項１に対応する発明と同様に作用する他、選択され得る展開対象のノードが複数ある場合、予め定められた戦略に基づき、実際に展開される展開対象が決定される。

【００３９】したがって、最も適切な展開戦略でもってプランニングを行うことができる。さらに、請求項３に対応する発明の階層的プランニング方法においては、請求項２に対応する発明と同様に作用する他、予め定められた戦略は、開始位置から介される基本的なアクションのノード数が最も少ない選択され得る展開対象を、実際に展開される展開対象として決定する戦略である。

【００４０】さらにまた、請求項４に対応する発明の階層的プランニング方法においては、請求項２に対応する発明と同様に作用する他、予め定められた戦略は、開始位置から介される基本的なアクションのノード数が最も多い選択され得る展開対象を、実際に展開される展開対象として決定する戦略である。

【００４１】一方、請求項５に対応する発明の階層的プランニング方法においては、請求項１～４に対応する発明と同様に作用する他、アクションやゴールが展開されることにより関連するノードでの状態が変化したときに、そのノードにおける状態の変化分が当該ノードでの状態に代えて保存される。

【００４２】したがって、全状態を一々保存する必要なく、ハードウェア資源の有効利用を実現できる。また、請求項６に対応する発明の階層的プランニング方法においては、請求項１に対応する発明と同様に作用する他、手続きネットワーク上におけるアクションやゴールのノードを展開する場合に、その展開に適用する展開知識が欠如しているとき、予め与えられた基本的なアクションの情報が用いられ、試行錯誤による探索で当該アクションやゴールを達成する基本的なアクションの系列が求められる。

【００４３】したがって、展開知識等の知識が完全ではなく抜けがある場合にもプランニングを続行することができる。さらに、請求項７に対応する発明の階層的プランニング方法においては、請求項６に対応する発明と同様に作用する他、探索で求められた基本的なアクションの系列，その探索開始時の状態及び対応するアクションやゴールの情報を保存し、アクションやゴールと同様なノードについて探索開始時と同じ状態なったとき、再び探索することなく探索で求められた基本的なアクションの系列が展開される。

【００４４】さらにまた、請求項８に対応する発明の階層的プランニング方法においては、手続きネットワーク上のアクションやゴールに対する展開知識におけるアクションやゴールの状態が、複数の状態述語によって表現される。

【００４５】ここで、状態術語は、状態を記述するものである。したがって、例えば一つのゴールに対して、その問題世界を構成する複数のオブジェクトについての状態を表現することが可能になる。

【００４６】このようにすることにより、展開知識を記述する際には、複数のオブジェクトについての状態を把握しつつ展開知識の記述が行われる。したがって、階層型プランナ制作者は、展開知識を容易に記述できることとなる。

【００４７】また、問題によっては、複数の状態術語をアクションやゴールの状態表現に用いることにより解き易くなる。一方、請求項９に対応する発明の階層的プラ

ンニング方法においては、請求項8に対応する発明と同様に作用する他、手続きネットワーク上のアクションやゴールを展開するとき、その展開対象のノードとしては、開始位置から基本的なアクションのノード及び又は補助的なノードのみを介して接続されるアクションやゴールのノードが選択される。

【0048】また、請求項10に対応する発明の階層的プランニング方法においては、請求項1又は8に対応する発明と同様に作用する他、手続きネットワーク上のノードが、状態述語によって定義された禁止状態に陥っているか否かが監視される。したがって、制約条件として禁止された状態を効果的に除外することができる。

【0049】

【発明の実施の形態】以下、本発明の実施の形態について説明する。図1は本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナの一例を示す構成図である。

【0050】この階層型プランナは、計算機システムにより実現されており、記憶装置に格納された知識ベース10と、この知識ベース10内のゴール展開知識10a及びアクション展開知識10bをコンパイルする処理装置内の知識コンパイラ11と、知識コンパイラ11にコンパイルされ、記憶装置内に保存されたプランナーエンジンの利用可能情報12と、入力装置から初期状態及び目標状態14を与えられるとプランナーエンジンの利用可能情報12を用いてプラン15を作成するプランナーエンジン13とによって構成されている。

【0051】次に、以上のように構成された本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナの動作について説明する。まず、知識ベース10には、ユーザが記述した、手続きネットワーク上のゴールノードとアクションノードを展開するためのゴール展開知識とアクション展開知識が格納される。

【0052】この知識ベース10に格納された知識は、知識コンパイラ11によってプランナーエンジン13の利用可能な形式に変換される。プランナーエンジン13に、初期状態と目標状態が入力されると、知識ベース10の知識を適用しながら手続きネットワークを展開し、プラン15を生成する。

【0053】この時、プランナーエンジン13によるゴールノード等の展開は、手続きネットワーク上で開始ノードからプリミティブアクションノードと補助ノードのみで結ばれたゴールノードについてのみなされる。すなわち図2に示すように、そのゴールノードまではプランが生成されているゴールノードのみを展開対象ノードとし、展開知識が適用可能ならば展開を実行することとする。

【0054】図2は本実施の形態の手続きネットワークにおける展開対象ノードを示す図である。同図に示すように、手続きネットワーク上で、開始ノード20からプリミティブアクションノード21と補助ノードのみ（この例ではｏｒ分岐／結合ノード23）で結ばれたゴールノード、すなわちそのノードまではプランが生成されているゴールノードのみ展開対象ゴールノード24としている。

【0055】ここで、開始ノード20には、初期状態が与えられているのでプリミティブアクションノード21の状態は確定していることになり、プランナーエンジン13は、プリミティブアクションノード21のアクション結果である確定状態をもとにゴール展開を行う。

【0056】したがって、展開対象ゴールノード24がプリミティブアクションノードに展開される場合には、少なくともここまでの過程で無駄なアクションを生じることなく、確実なゴールノード展開がなされることになる。また、このようにして展開されたプリミティブアクションノードもそのアクションにより確定的な状態を提供できることとなる。これにより、確実なゴールノード、アクションノード展開を実現することができる。

【0057】また、図2に示す手続きネットワークには、展開対象ゴールノードが複数あるが、このような場合の展開戦略について図3を用いて説明する。図3は本実施の形態の手続きネットワークにおける展開戦略を説明するための図である。

【0058】同図に示すように、手続きネットワーク上に展開対象ゴールノード34，35が複数ある場合、どのような位置にある展開対象ゴールノードから展開するかの戦略ルールがプランナーエンジン13にあらかじめ設定されている。ここで考えられる戦略ルールとしては、例えば、開始ノード30から最も近い展開対象ゴールノード34を優先的に展開する横型探索的な展開戦略と、開始ノード30から最も遠い展開対象ゴールノード35を優先的に展開する縦型探索的な展開戦略とが考えられる。

【0059】したがって、横型探索的な戦略ルールが設定された場合には、まず展開対象ゴールノード34が展開され、縦型探索的な戦略ルールが設定された場合は、まず展開対象ゴールノード35が展開される。

【0060】なお、設定し得る展開戦略ルールは、横型探索的な戦略と、縦型探索的な戦略とに限定されるものでなく、さらに例えば、横型探索的展開と縦型探索的展開とを交互に繰り返す複合型展開戦略など種々の戦略ルールが考えられる。

【0061】次に、各ノードにおける状態保持方法について説明する。必ずしも開始ノードからゴールノード等を展開することのない従来のプランナでは、プリミティブアクションは状態を変化させるものであるから、プリミティブアクションが実行されると実行前は成立していなかった状態が成立するようになったり、逆に成立していた状態が成立しなくなったりすることが生じる。従って、従来のプランナでは、手続きネットワーク上ではゴ

ールノードの展開などによって、プリミティブアクションノードが生じた場合、展開されたノード以降の全てのノードについて成立状態を変更する必要がある。

【００６２】本実施形態に係る階層型プランナでは、開始ノード側から順々に展開が進み、開始ノード側から順に状態が確定していくから、全てのノードに成立状態をあらかじめ登録しておく必要はない。そこで、本実施形態の階層型プランナでは、開始ノードからプリミティブアクションノードに展開されたところまでの各プリミティブアクションノードの状態変化の差分情報のみを登録する。

【００６３】図４は本実施形態において展開対象ノードで成立状態を得る方法を説明するための図である。同図（ａ）に示すように、開始ノード４０から展開対象ノード４３までのプリミティブアクションノード４１に関し、状態変化の差分情報が保存されている。

【００６４】ここで、展開対象ゴールノード４３は、展開対象ノードに展開知識が適用可能かどうかを調べるためには、展開対象ノードでの成立状態を知る必要がある。それには、状態がわかっているノードまで遡り、そこから差分情報を適用すると展開対象ノードの成立状態がわかる。

【００６５】そこで、図４（ａ）の場合、成立状態の参照が必要になった時には、開始ノード４０まで戻って開始ノード４０の初期状態と、成立状態がわかっているノード４１との差分情報とから当該展開対象ゴールノード４３の成立状態を導くことになる。

【００６６】この場合、遡らなければならないノードは少なく、状態更新の処理が大幅に短縮される。また、各成立状態を全て保存する場合に比べ、必要な記憶量が少ないので、ハードウェア資源の効率的使用を実現できる。

【００６７】また、同図（ｂ）に示すようにゴールノードの展開が進み、開始ノード４０から展開対象ゴールノード４４までのノード数が多くなる場合には、同図に示すように適宜のプリミティブアクションノード４３ｂにおける成立状態を保存し、展開対象ゴールノード４４から遡るノード数を少なくする。この場合では、プリミティブアクションノード４３ｂまで戻ればよい。

【００６８】次に、ゴールノードが既存の知識では展開できない場合、すなわち適切なゴール展開知識が用意されていない場合について説明する。このような場合には、ゴールノードの前のノードでの成立状態から出発し、プランナーエンジン１３は、プランナエンジンの利用可能情報１２の中から、その状態で実行可能なプリミティブアクションを適用していく。なお、プランナエンジンの利用可能情報１２には、プリミティブアクションの適用するためのオブジェクト知識、状態術語知識、アクション知識等が保存されている。

【００６９】プリミティブアクション適用の結果、状態が変化するので、その状態で実行可能なプリミティブアクションをさらに順次適用して行く。そしてゴールノードの状態が達成されればゴールノードをそのプリミティブアクションの系列と置き換える。

【００７０】実行可能なプリミティブアクションが複数あった場合は、どれかを選び適用する。探索に失敗した場合には、他のプリミティブアクションを適用する。探索の失敗は実行可能なプリミティブアクションが無くなった場合、一定のプリミティブアクションを適用してもゴールに到達しなかった場合などである。

【００７１】このときの様子を図５に示す。図５は本実施形態におけるゴールを展開するための探索の様子を示す図である。

【００７２】同図の展開対象ゴールノード５４については、プリミティブアクションノード５２の状態で知識ベース１０，つまりプランナエンジンの利用可能情報１２に適用可能なゴール展開知識が無いとする。

【００７３】したがって、プランナーエンジン１３は、プリミティブアクションノード５２の状態で実行可能なプリミティブアクションを実行したとして探索を行う。図５に示すように、展開対象ゴールノード５４の状態を達成するプリミティブアクションノード系列５３を発見できた場合、このノード系列５３を展開対象ゴールノード５４と置き換える。

【００７４】また、このように、探索によって展開対象ゴールノード５４のプリミティブアクションノード系列５３への置き換えが起こった場合、展開対象ゴールノード５４のゴールパターン，その直前のプリミティブアクションノード５２での成立状態及び探索によって得たプリミティブアクション系列５３の３つをプランナーエンジン１３の内部データとして記憶する。

【００７５】これにより、以後のプランニングの過程で、同じ状態で同じゴールノードを展開する時は、探索を実行すること無く、ゴールノードとプリミティブアクション系列の置き換えを実行する。

【００７６】さらに、図５に示す場合に、探索によりプランナーエンジンの内部データとして記憶された上記３つの情報は、プランニングが終了するとその記憶が失われてしまうため、記憶したプリミティブアクション系列についての情報を通常の展開知識と同じ形式で記憶装置内のファイルに出力する。

【００７７】このようにして、プリミティブアクションノード５２の成立状態を展開条件、展開対象ゴールノード５４のゴールパターンをゴールパターン、プリミティブアクションノード系列５３を展開内容とする新たな展開知識を、知識ベース１０にて、次のプランニング時にも利用できるようにする。

【００７８】次に、ゴールパターンの状態術語の取り扱いについて説明する。また、従来の階層型プランナにおいては、ゴールパターンは必ず１つの状態述語で表現さ

れていたが、本実施形態においては、ゴールパターンを複数の状態述語による表現を可能とする。

【００７９】つまり、本実施形態では、例えばプランの対象世界を構成するオブジェクトが3つのオブジェクトA，B，Cから構成されている場合には、本来的にはオブジェクトA，B，Cすべてについて状態が存在していることに着目し、あるノードにおいて、そのノードのアクションに係わる状態のみでなく、例えばオブジェクトA，B，Cの各状態を複数の状態術語を使って表現できるようにしている。

【００８０】したがって、例えば展開内容に複数述語表現のゴールノードをサブゴール展開する場合のゴール展開知識を記述する場合には、展開内容のゴールノードのゴールパターンとして複数の状態述語を記述する。

【００８１】また、同様に、複数述語表現のゴールノードをプリミティブアクションノードによるアクションに展開するためのアクション知識を記述する場合にも、展開の対象となるゴールパターンとして複数の状態述語を記述する。

【００８２】このように複数の状態を記述することで、プランナ制作者は、各オブジェクトの状態を認識しつつ展開知識を記述できることになり、展開知識の作成が極めて容易となる。また、取り扱う問題によっては、このように複数の状態述語を用いた方が問題を解き易くなる。

【００８３】なお、プランナーエンジン１３は、複数の状態述語のゴールパターンを持つゴールノードを展開する場合も、ゴールパターンが一つの状態述語で表現されたゴールノードを展開する場合と同じように、ゴールパターンが同じ展開知識のうち展開条件が成立している展開知識を適用する。

【００８４】次に、プラン作成上の制約条件として禁止された状態がある場合について説明する。このような場合には、その禁止状態を状態述語のリストとして、予めファイルに登録しておく。そして、プランナーエンジンは、この禁止状態を読み込み、ノードの展開または探索によって新たなプリミティブアクションノードが生成され、手続きネットワーク上のノードでの成立状態が変わると、その都度、その状態が禁止状態となっているか否かをチェックし、禁止状態になる場合はその操作を中止、すなわち直前の展開または探索を取り消す。

【００８５】以上のような処理を経て、全てのゴールノードの展開が終了し、手続きネットワーク上がプリミティブアクションノードと補助ノードのみとなったら、相互作用の解消を行う。相互作用の検出解消の方法は、従来技術で説明した通りである。

【００８６】そして、相互作用となる場合のプリミティブアクションの組を検出したときは、適切な順序付けを行う。相互作用を解消するような順序付けができない場合は、再プランニングする。

【００８７】上述したように、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、開始ノード側から展開が進むようにしたので、無駄なアクションを含んだプランの生成が抑制される。従って、全体のプラン生成効率も向上する。

【００８８】すなわち開始ノードに最も近いゴールから順に展開し、かつ、変数でなく初期状態から順に確定する状態でもってゴールをプリミティブアクションノードに展開することで、無駄なアクションを含むプランの生成が抑制され、かつ、どのような問題に対しても非常に高い確実性をもってプランを生成できる。

【００８９】このとき、展開の自由度が従来方法ほど高くないことから、展開知識の作成難度が低くなり、簡単かつ確実な階層型プランナを作成することができる。また、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、問題に適する展開戦略を選ぶことにより効率的なプラン生成が可能となる。また、展開対象ノードがＡＮＤ並列の場合は横型探索、ＯＲ並列の場合は縦型探索等のように設定することで無駄な展開を抑制することもできる。

【００９０】さらに、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、展開に伴う各ノードでの成立状態の更新処理を必要な時まで遅らせるようにしたので、更新処理の時間を大幅に短縮することができる。また、成立状態の更新が必要な場合も展開対象ノードより開始ノード側のノードのみ更新するので処理時間が短くなる。

【００９１】また、適宜なノード位置で成立状態を保存するようにしたので、より一層更新処理の時間を大幅に短縮することができる。さらにまた、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、展開知識が完全ではなく抜けがある場合や展開知識の記述が困難な場合にも探索によってプランニングが実行できるようにしたので、不完全な知識ベースで、不十分な知識でもプランニングが可能となり、これによりプランナの適用範囲が広がる。

【００９２】また、展開知識が完全ではなく抜けがある場合や展開知識の記述が困難な場合にも、以前の探索結果を保存しているので、同じ状況では再び探索することなく展開することができ、同じ探索を２回以上行わないのでプランの生成効率を上げることができる。

【００９３】さらに、上記探索を行った場合に、プランニングが終了しても探索結果を破棄せずファイルに保存するようにしたので、次回のプランニングからは同じ探索を繰り返さずプランニングの効率が向上する。

【００９４】また、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、意味のある状態を１つの述語では表せない場合に、複数の述語を組み合わせるようにしたので、かかる場合でもゴール状態を表現でき、展開知識等の知識の記述力を増す

ことができる。したがって、階層型プランナの適用範囲を広げることができる。

【００９５】さらに、本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナによれば、ユーザは禁止状態を禁止状態のリストに登録するようにしたので、他は禁止状態を意識して知識を記述する必要を無くすことができる。このためユーザの知識記述の負担が減少する。また、禁止状態をすぐに検出できるので無駄な探索を省くことができる。なお、本発明は、上記各実施の形態に限定されるものでなく、その要旨を逸脱しない範囲で種々に変形することが可能である。

【００９６】

【実施例】以下、本発明の実施例について説明する。図６は本発明の実施例に係る階層的プランニング方法を適用した階層型プランナの一例を示す構成図である。

【００９７】この階層型プランナは、計算機システムによって実現されており、また、知識ベース６０を格納する知識ベース記憶装置６１と、この知識ベース６１内の各知識をプロダクションシステム６８で処理可能なルール形式に変換する処理装置６２内の知識コンパイラ６３と、知識コンパイラ６３にコンパイルされたプロダクションルール６４を格納するルール記憶装置６５と、入力装置から初期状態及び目標状態６６を与えられるとプロダクションルール６４を用いてプラン６７を作成する処理装置６２内のプロダクションシステム６８とによって構成されている。

【００９８】また、知識ベース６０には、ゴールノードを展開するための知識であるゴール展開知識６０ａと、アクションノードを展開するための知識であるアクション展開知識６０ｂと、オブジェクト知識６０ｃと、状態術語知識６０ｄと、アクション知識６０ｅとが保存されている。

【００９９】一方、プロダクションシステム６８には、プロダクションシステムを利用して動くプランナエンジン６８ａと、ワーキングメモリ６８ｂとが設けられている。このように構成された本実施例の階層型プランナを用い、ブロックワールド問題を例として取り扱った場合を説明する。

【０１００】ブロックワールド問題はプランニング問題の一つであり、十分に広いテーブルの上に置かれたブロックを別の状態に置き換える問題である。ただし、ブロックは一度に一つしか移動できない。典型的な例を図７に示す。

【０１０１】図７は本実施例におけるブロックワールドの初期状態と目標状態を示す図である。同図のような初期状態７０から目標状態７１に移行するためには以下のようにブロックを移動すればよい。

＊【０１０２】１．ブロックＣをテーブルの上に置く。
２．ブロックＢをブロックＣの上に置く。
３．ブロックＡをブロックＢの上に置く。

【０１０３】このような手順を自動的に求めるのがこのプランニングの目的である。このプランニングにおいては、まず対象世界を構成するオブジェクトを定義する必要がある。この内容はオブジェクト知識６０ｃとし、知識ベース記憶装置６１内の知識ベース６０の一部として格納する。ブロックワールドにおいては、オブジェクト知識６０ｃは例えば以下のように記述する。

【０１０４】

```
ｂｌｏｃｋ　Ａ
ｂｌｏｃｋ　Ｂ
ｂｌｏｃｋ　Ｃ
ｔａｂｌｅ　Ｔ
```

次に、ブロックの位置関係を表すために状態述語を定義する必要がある。これを状態述語知識６０ｄとして知識ベース記憶装置６０に格納する。状態述語知識６０ｄの記述例を以下に示す。

【０１０５】

```
ｏｎ（ｂｌｏｃｋ，ｂｌｏｃｋ）
ｏｎｔａｂｌｅ（ｂｌｏｃｋ）
ｃｌｅａｒ（ｂｌｏｃｋ）
```

ここで、ｏｎは、ブロック型のオブジェクト２つを引数に取り、ブロックが他のブロックの上にあることを表している。また、ｏｎｔａｂｌｅとｃｌｅａｒはブロック型のオブジェクトを１つ引数に取り、それぞれブロックの上には何もないこと、ブロックがテーブルの上にあることを表している。これらの状態述語を用いて初期状態と目標状態は次の様に表される。

【０１０６】

初期状態	目標状態
ｏｎ（Ｃ，Ａ）	ｏｎ（Ａ，Ｂ）
ｏｎｔａｂｌｅ（Ａ）	ｏｎ（Ｂ，Ｃ）
ｏｎｔａｂｌｅ（Ｂ）	ｏｎｔａｂｌｅ（Ｃ）
ｃｌｅａｒ（Ｂ）	ｃｌｅａｒ（Ａ）
ｃｌｅａｒ（Ｃ）	

さらに、状態を変化させる操作としてアクションを定義し、アクション知識６０ｅとして知識ベース記憶装置６０に格納する。

【０１０７】ここでは、「ブロックを他のブロックの上に置く。」というｐｕｔ１と、「テーブルの上のブロックをブロックの上に置く。」というｐｕｔ２と、「ブロックの上のブロックをテーブルの上に置く。」というｔｏｔａｂｌｅの３種類のアクションをプリミティブアクションとして定義する。アクション知識の記述例を以下に示す。

```
ｐｕｔ１：
    primitive       put1     (block x, block y)
        precondition  clear(x) clear(y) on(x,z)
```

```
        add             on(x,y)
        delete          clear(y) on(x,z)
put2:
    primitive       put2    (block x, block y)
        precondition    clear(x) clear(y) ontable(x)
     add            on(x,y)
        delete          clear(y) ontable(x)
totable:
    primitive       totable  (block x)
        precondition    clear(x) on(x,z)
        add             clear(z) ontable(x)
        delete          on(x,z)
```

アクションput1は、変数であるところのブロック型のオブジェクトの引数x，yを持ち、アクションを実行するための前提条件はclear（x） clear（y） on（x，z）で、アクションの結果on（x，y）が成立し、clear（y） on（x，z）が成立しなくなることを表している。

【0108】次に、目標をより具体的な副目標やアクションに分解するためのゴール展開知識60a及びアクション知識60bを記述し、知識ベース記憶装置60に格納する。ゴール展開知識60aの記述例を以下に示す。

【0109】ゴール展開知識その1：

```
goal  pattern  clear(x)
when  on(y, x)
into  totable(y)
```

ゴール展開知識その2：

```
goal  pattern  on(x, y)
when  on(x, z)
into  put1(x, y)
```

ゴール展開知識その3：

```
goal  pattern  on(x, y)
when  ontable(x)
into  put2(x, y)
```

ここで、ゴール展開知識その1は、clear（x）というゴールパターンを持つゴールノードは、そのノードでon（x，y）が成立している時、totable（y）に分解できることを表している。ここでは、ゴールノード1つが1つのプリミティブアクションに置き換わっているが、一般的には複数のノードから成るサブネットワークに置き換えられる。

【0110】このブロックワールドの例では、上記目標状態より、on（A，B）、on（B，C）、ontable（C）、clear（A）が達成できればよいから、上記ゴール展開知識及びアクション知識を用いれば、ゴールを展開し、プランを作成できることがわかる。

【0111】また、以上より、各知識はプロダクションルールの形式で入力するよりも簡単な表現で記述されることがわかる。次にこのような各知識を与えられた階層＊型プランナにおけるプラン作成動作について説明する。

【0112】まず、知識コンパイラ63は、オブジェクト知識60c、状態述語知識60d、アクション知識60eを参照しながらゴール展開知識60aをプロダクションルール64に変換する。プロダクションルール64はルール記憶装置65に格納される。

【0113】一方、プロダクションシステム68上では、まず、初期状態及び目標状態66がワーキングメモリ68bに読み込まれる。以降、手続きネットワークの各ノードはワーキングメモリ68b内の1つのWM要素（ワーキングメモリ要素）として表現され、各WM要素に親ノードと子ノードが登録されることで手続きネットワークが表現される。プランナーエンジン68aは、このために必要な処理を行う。

【0114】例えば以下のようなWM要素が用いられる。

（要素名，int ノードID属性，int パターンID属性，int ノードタイプ属性，int set 親ノード，int set 子ノード）

注：int は整数値型，int set は整数値の集合型を意味する。

【0115】また、状態述語知識も例えば以下のようなWM要素を用いて表わされる。

（on　　，int ファクトID，string引数1，string引数2）；

（ontable，int ファクトID，string引数1）；

（clear　，int ファクトID，string引数1）；

注：stringは文字列型を意味する。

【0116】上記ゴール展開知識その1をプロダクションルール64に変換した例を概念的に示すと次のようになる。

if

ゴールノードがある。

【0117】

ゴールパターンがclear（x）である。

展開条件を満たしている。

then

新しいノードを表すWM要素を生成する。

【0118】
親ノード、子ノードを設定する。
各新ノードでの状態の成立状況の設定する。
ここで、プランナーエンジン68aは、初期状態及び目標状態66が入力され、これを受け取ると、開始ノード、目標状態を表すゴールノード、終了ノードからなる手続きネットワークをプロダクションシステム68のWM要素を用いて生成する。
【0119】次に、ゴールノードは、プロダクションルール64に変換された展開知識を用いてプランナーエンジン68aによって展開される。展開方法は発明の実施の形態で説明した方法と同様である。
【0120】まず、開始ノードからプリミティブアクションまたはファントムゴールまたは補助ノードのみで結ばれたゴールノードのみを展開対象ゴールノードとする。また、展開対象となるゴールノードが複数ある時は、開始ノードから遠いゴールノードを優先的に展開する（縦型探索）、又は開始ノードに近いゴールノードを優先的に展開する（横型探索）の何れかを選択するかが、初期設定において設定されている。
【0121】さらに、プリミティブアクションノードにプリミティブアクションによる効果のみが登録されており、直前のノードの状態が確定したらその効果を反映して新しい状態を導出することで、対象となるノードに適用する状態を求める。
【0122】一方、展開対象ゴールノードに適用できる展開知識がない場合にはプランナーエンジン68aは以下のような処理を起動する。例えば、図8に示すようにあるプリミティブアクション80の次に、ゴールパターンがｏｎｔａｂｌｅ（Ａ）であるような展開対象ノード（ゴールノード81）があるとする。
【0123】図8は本実施例におけるアクションの探索を説明するための図である。ここで、プリミティブアクションの結果、状態は図8に示すようになっており、ｏｎｔａｂｌｅ（Ａ）を達成するための展開知識はないとする。
【0124】すると、プランナーエンジン68aにより探索が起動される。探索機能は、図8に示す状態で実行可能なプリミティブアクションを調べる。ここでは、以下に示す3つのアクションが実行可能であるとする。

＊アクション　追加リスト　　削除リスト
ptu1(A,C)　clear(B) on(A,C)　clear(C) on(A,B)
ptu2(C,A)　on(C,A)　　clear(A) ontable(C)
totable(A) clear(B) ontable(A) on(A,B)

この3つのアクションのうち、ｔｏｔａｂｌｅ（Ａ）が追加リストにｏｎｔａｂｌｅ（Ａ）を持つことからこのプリミティブアクションを用いれば、ゴールノード81は展開できることがわかる。そこで、プランナーエンジン68aは、このプリミティブアクションを選択し、探索を終了する。
【0125】もし、追加リストの中にｏｎｔａｂｌｅ（Ａ）を持つアクションがなければ、プランナーエンジン68aは、任意のアクションを選び、追加／削除リストを反映して状態を更新し、再度実行可能なアクションを求める。この操作を一定の回数繰り返す。それでもｏｎｔａｂｌｅ（Ａ）が達成されなければ、バックトラックして他のアクションを試す。バックトラックを繰り返し、全ての経路を探索してもｏｎｔａｂｌｅ（Ａ）が達成されなければ、探索を終了する。
【0126】また、この探索結果は、ワーキングメモリ68bに記憶される。具体的に記憶される内容は、状態述語のリストで表現された探索開始時の状態、状態述語で表現されたゴールパターン、探索で得たプリミティブアクションの系列である。探索が起動したら、プランナーエンジン68aは、まず、この記憶を調べ、同じ状態の同じゴールを持つものがある場合、探索をせずに過去の例を用いて展開を行う。
【0127】さらに、ワーキングメモリ68bに記憶された上記探索結果は、ゴール展開知識の形式でファイルに出力される。ここで、探索開始時の状態は展開条件、プリミティブアクションの系列は展開内容となる。ゴールパターンはそのままゴールパターンとする。
【0128】また、ゴールパターンは、複数の述語を記述することを可能とし、ゴール展開知識もそれに応じて複数の述語を記述するようにすることを可能とする。例えば、ｏｎｔａｂｌｅ（ｘ）とｃｌｅａｒ（ｘ）の2つの述語をゴールパターンに持つゴールノードを展開する知識は次のように記述する。
【0129】

```
goal pattern ontable(x) clear(x)
    when ...
    into ...
```

ここで、ｗｈｅｎ（展開条件）とｉｎｔｏ（展開内容）は上記例と同様に記述する。
【0130】この知識をコイパイルしたプロダクションルール64では、複数の述語に対応して、条件部にゴールパターンの数だけ条件式を増やせばよい。また、展開※

※内容にｏｎｔａｂｌｅ（ｘ）とｃｌｅａｒ（ｘ）の2つの述語をゴールパターンに持つゴールノードを記述するには次のように複数述語を［と］でくくる。
【0131】

```
goal pattern xxxx(x)
    when ...
```

into [ontable(x), clear(x)]

さらに、禁止状態のチェックは、禁止状態が状態述語のリストとして表現され、記憶されることにより実現される。展開または探索が起こり、各ノードの状態が更新されたらその状態が禁止状態になっていないかどうかを比較して調べ、禁止状態になっていれば、その展開または探索を取り消す。

【0132】このようにして、展開が終了し、全てのノードがプリミティブアクションノードまたは補助ノードになった後、プランナーエンジン63は相互作用の検出と解消を行う。

【0133】相互作用は、順序付けされていない並列なプリミティブアクション間で、その前提条件と効果が相反するものを探して検出する。相互作用を検出した場合は、前提条件が壊されてしまうプリミティブアクションを先に、他方のプリミティブアクションを後に順序付けする。全ての相互作用が解消できたら、プラン67を出力する。解消できない相互作用があった場合は、再プランニングする。

【0134】

【発明の効果】以上詳記したように本発明によれば、無駄なアクションを含むプランの生成を抑制し、同時に全体のプラン生成効率を向上させる階層的プランニング方法を提供することができる。

【0135】また、本発明によれば、知識が完全ではなく抜けがある場合にもプランニングを続行できるようにする階層的プランニング方法を提供することができる。さらに、本発明によれば、展開知識の記述しづらい問題にも対応することができる階層的プランニング方法を提供することができる。一方、本発明によれば、制約条件として禁止された状態を効果的に除外することを可能とする階層的プランニング方法を提供することができる。

【図面の簡単な説明】

【図1】本発明の実施の形態に係る階層的プランニング方法を適用した階層型プランナの一例を示す構成図。

【図2】同実施の形態の手続きネットワークにおける展開対象ノードを示す図。

【図3】同実施の形態の手続きネットワークにおける展開戦略を説明するための図。

【図4】同実施形態において展開対象ノードで成立状態を得る方法を説明するための図。

【図5】同実施形態におけるゴールを展開するための探索の様子を示す図。

【図6】本発明の実施例に係る階層的プランニング方法を適用した階層型プランナの一例を示す構成図。

【図7】同実施例におけるブロックワールドの初期状態と目標状態を示す図。

【図8】同実施例におけるアクションの探索を説明するための図。

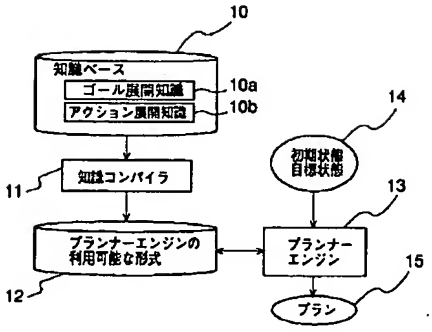
【図9】階層型プランナを用いてプラン候補を作成する例を示す図。

【図10】階層型プランナにおける相互作用のチェックを説明するための図。

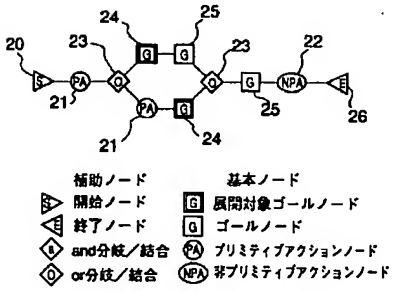
【符号の説明】

10…知識ベース
10a…ゴール展開知識
10b…アクション展開知識
11…知識コンパイラ
12…プランナーエンジンの利用可能情報
13…プランナーエンジン
14…初期状態及び目標状態
15…プラン
20，30，40，50…開始ノード
21，31，41，42，51，52，80…プリミティブアクションノード
22，32…非プリミティブアクションノード
23，33…or分岐／結合ノード
24，34，35，43，54…展開対象ゴールノード
25，36，44，55，81…ゴールノード
26，37，45，56…終了ノード
53…プリミティブアクションノード系列
60…知識ベース
61…知識ベース記憶装置
62…処理装置
63…知識コンパイラ
64…プロダクションルール
65…ルール記憶装置
66…初期状態及び目標状態
67…プラン
68…プロダクションシステム
68a…プランナーエンジン
70…初期状態
71…目標状態

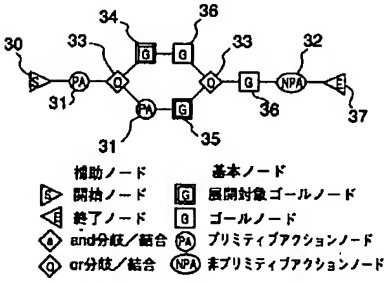
【図1】



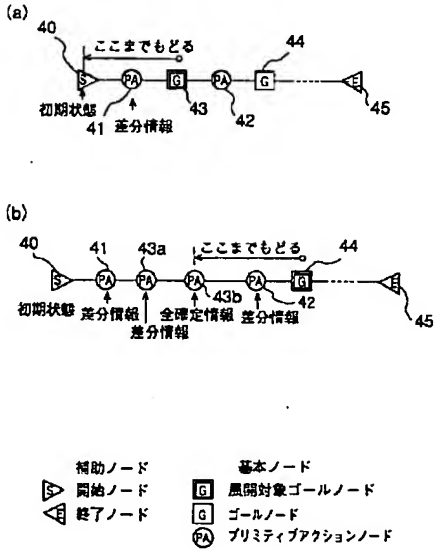
【図2】



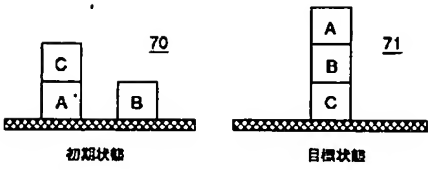
【図3】



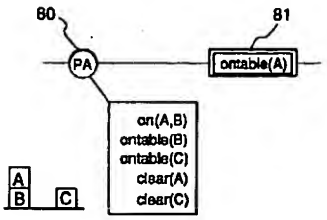
【図4】



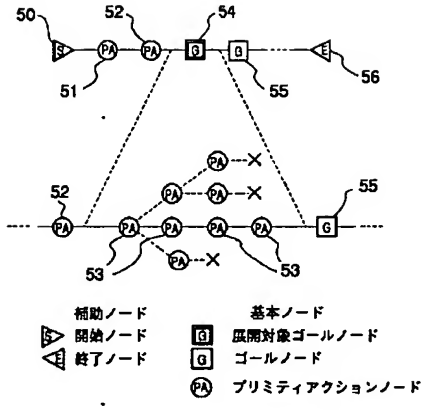
【図7】



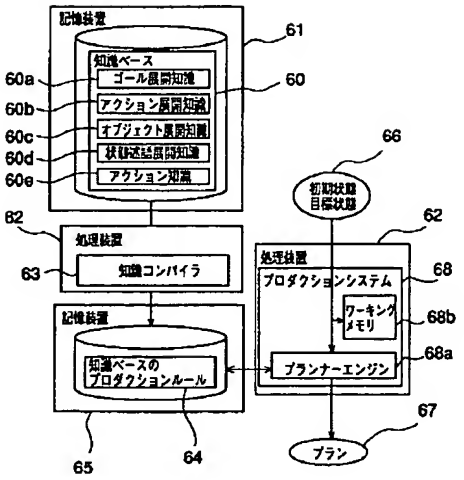
【図8】



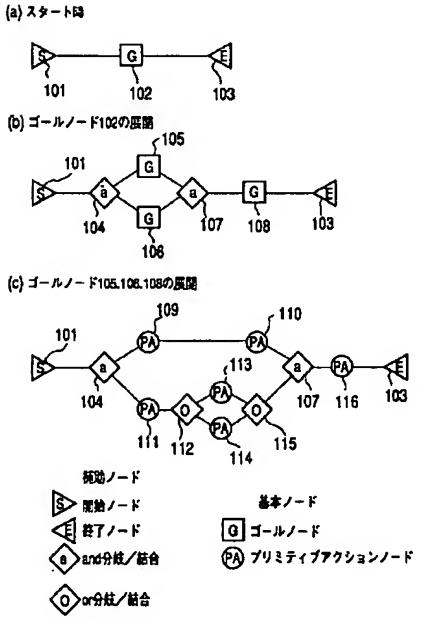
【図5】


50
52
54
51
55
56
52
55
53
53
補助ノード
開始ノード
終了ノード
基本ノード
展開対象ゴールノード
ゴールノード
プリミティブアクションノード

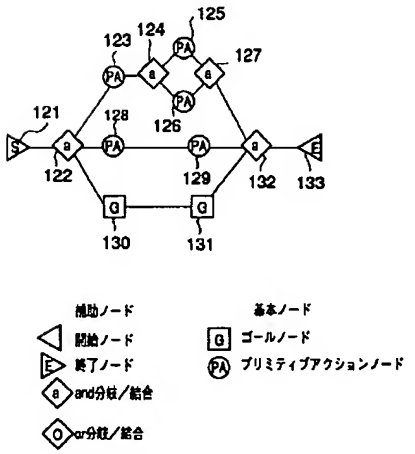
【図6】


記憶装置
61
知識ベース
ゴール展開知識
アクション展開知識
オブジェクト展開知識
状態遷移展開知識
アクション知識
60
60a
60b
60c
60d
60e
66
初期状態
目標状態
62
62
処理装置
知識コンパイラ
63
処理装置
プロダクションシステム
68
ワーキングメモリ
68b
プランナーエンジン
68a
記憶装置
知識ベースのプロダクションルール
67
プラン
65
64

【図9】


(a) スタート時
101
102
103
(b) ゴールノード102の展開
105
101
104
106
107
108
103
(c) ゴールノード105,106,108の展開
109
110
101
104
113
107
116
103
111
112
114
115
補助ノード
開始ノード
終了ノード
and分岐/結合
or分岐/結合
基本ノード
ゴールノード
プリミティブアクションノード

【図10】


125
123
124
127
121
128
126
122
129
132
133
130
131
補助ノード
開始ノード
終了ノード
and分岐/結合
or分岐/結合
基本ノード
ゴールノード
プリミティブアクションノード

Notes:
1. Untranslatable words are replaced with asterisks (****).
2. Texts in the figures are not translated and shown as it is.

Translated: 04:52:06 JST 06/09/2006
Dictionary: Last updated 05/26/2006 / Priority: 1. Information communication technology (ICT) / 2. Electronic engineering / 3. Mathematics/Physics

---

FULL CONTENTS

[Claim(s)]

[Claim 1] By making fundamental action into sequential execution to the initial state in a starting position It is the method of generating as a plan the sequence of action which changes a state from said initial state to a goal state. In the hierarchical planning method of obtaining said plan by developing the procedure network which makes a node action connected one by one from said starting position, and Gaul for the sequence which consists of said fundamental action When developing said action on said procedure network, and Gaul, as a node for [the] expansion the node of said starting position to said fundamental action -- and -- or the hierarchical planning method characterized by choosing said action connected only through an auxiliary node, and the node of Gaul.

[Claim 2] The hierarchical planning method according to claim 1 characterized by determining the candidate for expansion actually developed based on the strategy defined beforehand when there are two or more nodes for [which may be chosen] expansion.

[Claim 3] Said strategy defined beforehand is the hierarchical planning method of ******2** characterized by the node number of said fundamental action minded from said starting position being the strategy of determining said fewest candidates for expansion that may be chosen as said candidate for expansion actually developed.

[Claim 4] Said strategy defined beforehand is the hierarchical planning method according to claim 2 characterized by being the strategy of determining said candidate for expansion with most node numbers of said fundamental action minded from said starting position which may be chosen as said candidate for expansion actually developed.

[Claim 5] It is the hierarchical planning method of any or given in 1 clause, among the Claims 1-4 characterized by replacing with and saving a changed part of the state in the node in the state in the node concerned when the state in the node related by developing said action and Gaul changes.

[Claim 6] When developing action on said procedure network, and the node of Gaul and the expansion knowledge applied to the expansion is lacking, The hierarchical planning method given in the Claim 1 clause characterized by searching for the sequence of the action concerned or fundamental action which attains Gaul by search by trial and error using the information on fundamental action given beforehand.
[Claim 7] At the time of the state ****** same about said action or the same node as Gaul as the time of a search start which saves the sequence of fundamental action required in said search, the state at the time of the search start, and corresponding action and the information on Gaul The hierarchical planning method given in the Claim 6 clause characterized by developing the sequence of fundamental action called for by said search without searching again.
[Claim 8] By making fundamental action into sequential execution to the initial state in a starting position It is the method of generating as a plan the sequence of action which changes a state from said initial state to a goal state. In the hierarchical planning method of obtaining said plan by using expansion knowledge for the sequence which consists of said fundamental action the procedure network which makes a node action connected one by one and Gaul, and developing from said starting position The hierarchical planning method characterized by expressing said action on said procedure network, action in the expansion knowledge over Gaul, and the state of Gaul by two or more stative predicates.
[Claim 9] When developing said action on said procedure network, and Gaul, as a node for [the] expansion the node of said starting position to said fundamental action -- and -- or the hierarchical planning method according to claim 8 characterized by choosing said action connected only through an auxiliary node, and the node of Gaul.
[Claim 10] The hierarchical planning method according to claim 1 or 8 characterized by supervising whether the node on said procedure network has lapsed into the inhibiting state defined by the stative predicate.

[Detailed Description of the Invention]
[0001]
[Field of the Invention] This invention relates to the hierarchical planning method which draws up the plan (plan) for determining the action plan generated in various fields.
[0002]
[Description of the Prior Art] Using the hierarchical planner by a computer about the plan creation for determining the action plan in recent years to the event generated in various fields is performed.

[0003] There is deciding the sequence that the power supply of a computer network should be dropped on an easy example etc. Moreover, after electric power system is downed, for example, it considers using this hierarchical planner as an expert system for drawing up the plan to decide the turn of the control device when restoring this.
[0004] Here, a plan means the sequence of action which changes a state from an initial state to a goal state, and means that a planning asks for the plan which can be performed when three, an initial state, a goal state, and action, are given.
[0005] There is the hierarchical planning method using the procedure network as one of the conventional methods of this planning. A procedure network is the internal expression of the plan of planning process, and consists of an auxiliary node which shows the start of the Gaul node, two kinds of basic nodes of an action node, and a plan, an end, etc. There are two of the un-primitive action nodes showing the primitive action node showing fundamental action and abstract action in an action node.
[0006] A goal state is expressed as the un-primitive action node and the Gaul node on a procedure network. A planning progresses the Gaul node and an action node by subnetwork replacement ****** which consists of a more concrete Gaul (subgoal) node or an action node. This replacement is called expansion of a node. Expansion is performed according to the knowledge given beforehand. The time of all the basic nodes becoming primitive action is a time of the candidate of a plan being obtained.
[0007] Finally the interaction between actions is detected, sequencing suitable between actions is carried out; and an interaction is canceled. If dissolution is impossible, it will re-plan. Here, an interaction means the case where the effect of one primitive action negates the prerequisite of primitive action of another side, between primitive actions which can be performed in parallel on a procedure network. If all interactions are cancelable, it is outputted as a final plan.
[0008] The easy example of the plan creation by this hierarchical planner is shown in drawing 9 . Drawing 9 is the figure showing the example which creates a plan candidate using a hierarchical planner. In this figure (a), the procedure network consists only of the start node 101, a Gaul node 102, and an end node 103 first at the time of a start. An initial state is given to the start node 101 and the goal state is expressed to it by the Gaul node 102. Moreover, if an initial state is processed by the contents of the Gaul node 102, a goal state will be attained and it will result in the end node 103.
[0009] In order to draw up a plan, this Gaul node 102 is first developed by the Gaul expansion knowledge which was able to be given beforehand. The Gaul node 102 is developed and a procedure network will be in the state by which it is shown in drawing 9 (b) by making it the concrete Gaul node 105,106,108.
[0010] If this Gaul node 105,106,108 each is further developed by the Gaul expansion

knowledge, it will be in the state which shows in drawing 9 (c). In this figure (c), all the Gaul nodes are developed and all basic nodes have become the primitive action node 109, 110, and 111,113,114,116. In addition, in order to connect these primitive action nodes, and branching / combination 104,107, and or branching / combination 112,115 are used.

[0011] In this completed plan, when the initial state of the start node 101 is sequentially processed by each node and it results in the end node 103, the final purpose state will be acquired. In addition, although the above-mentioned example showed only the Gaul node as a node which should be developed The node (an un-primitive action node, action node) of un-primitive action as well as the Gaul node [with expansion knowledge (action expansion knowledge)] It is developed by the primitive action node etc., and an un-primitive action node is used when dealing with contents more abstract than the case of the Gaul node as action.

[0012] Moreover, an interaction is checked in the gradual proper state in the middle of the plan candidate who completed, or expansion. As the example of drawing 10 R> 0 shows, an interaction may be produced when the primitive action node has been arranged in parallel by and branching / combination.

[0013] Drawing 10 is a figure for explaining the check of the interaction in a hierarchical planner. In the example of this figure, [an interaction] The primitive action node 123<-> primitive action node 128, the primitive action node 123<-> primitive action node 129, the primitive action node 125<-> primitive action node 126, The primitive action node 125<-> primitive action node 128, the primitive action node 125<-> primitive action node 129, the primitive action node 126<-> primitive action node 128, It checks between the primitive action node 126<-> primitive action nodes 129.

[0014] That is, it will be carried out to all the pairs of primitive action which can be performed simultaneously. Here, when it is necessary to set in order by detecting an interaction by some pairs, sequencing is performed so that all the interactions may be canceled.

[0015] By the way, above-mentioned drawing 9 aims the actual hierarchical planner at enabling creation of a very complicated plan, although the very easy example is shown. From such a view, while each Gaul node and an action node have had a variable as an argument, expansion is performed. That is, according to the rule which the planner equipment has, the proper Gaul node etc. is chosen and developed out of all the Gaul nodes in a certain stage, or an action node.

[0016] Therefore, when there is unsuitable expansion in the process which develops the above-mentioned Gaul node etc., unsuitable expansion is canceled, and a planning is performed, searching for redoing expansion etc.

[0017] Thus, the method of developing the proper Gaul node in the Gaul node in the stage has a possibility that it can search with high flexibility and a planning can be realized in the small amount of search, having a variable.

[0018]
[Problem to be solved by the invention] However, in the method of developing proper Gaul, with the conventional variable having, there is a problem that it is difficult to create expansion knowledge for a hierarchical planner producer to develop the Gaul node conversely since the flexibility of expansion is high.
[0019] Moreover, by this method, as described above, a planning may be realizable with the small amount of search, but since all the nodes that suit the conditions of expansion knowledge by one side are considering it as the node which can be developed, they may generate the plan under which useless action is included.
[0020] Although it is seldom unthinkable to be such in the case of an easy plan and actions X and Y cannot be carried out, unless it is materialized in the case of [A] a complicated plan (for example, some conditions) The node which carries out actions X and Y may be created before node creation of the action Z which satisfies Conditions A.
[0021] In such a case, after creating the action contrary Y, reverse X, and the node to turn back to the node creation time of the action Z of Conditions A, the node of Action Z will be created. Therefore, you have to set to Actions X and Y, Contrary Y, reverse X, Z, X, and Y essentially the place which ends by Action Z, X, and Y in this case.
[0022] That is, although the conventional method may be able to draw up an efficient plan in the small amount of search, it may draw up the plan under which useless action is included simultaneously. Therefore, it is difficult to draw up few useless plans certainly to any action plans generated in various fields. This causes [of the plan] a production efficiency fall.
[0023] Moreover, by the conventional method, when there was no knowledge which develops Gaul whose expansion knowledge is not perfect to case [Gaul] namely, attain, generation of the plan had gone wrong. On the other hand, although the object world of a plan is constituted by the object, action which one node performs is related with one object. Therefore, as for the state where it is obtained as a result of action to an object, only one is described in a basic node. However, by the conventional state representation method, there was a case where description of the knowledge which develops a goal state in the subgoal or action was difficult.
[0024] Moreover, by the conventional method, the state where it was forbidden as a constraint was not excepted effectively. This invention was made in consideration of such the actual condition, and the 1st purpose controls generation of the plan under which useless action is included, and there is in offering the hierarchical planning method which raises the whole plan production efficiency simultaneously.
[0025] Moreover, the 2nd purpose does not have perfect knowledge, and also when there is an omission, there is in offering the hierarchical planning method which enables it to continue a planning. Furthermore, the 3rd purpose is to offer the hierarchical planning method which can also cope with the problem expansion knowledge is hard to describe. On the other hand, the

4th purpose is to offer the hierarchical planning method which makes it possible to except effectively the state where it was forbidden as a constraint.

[0026]

[Means for solving problem] [invention corresponding to Claim 1] in order to solve the above-mentioned technical problem By making fundamental action into sequential execution to the initial state in a starting position It is the method of generating as a plan the sequence of action which changes a state from an initial state to a goal state. In the hierarchical planning method of obtaining a plan by developing the procedure network which makes a node action connected one by one from the starting position, and Gaul for the sequence which consists of fundamental action the time of developing action on a procedure network, and Gaul -- the node of action fundamental from a starting position as a node for [the] expansion -- and -- or it is the hierarchical planning method which chooses action connected only through an auxiliary node, and the node of Gaul.

[0027] Moreover, in invention corresponding to Claim 1, invention corresponding to Claim 2 is the hierarchical planning method of determining the candidate for expansion actually developed based on the strategy defined beforehand, when there are two or more nodes for [which may be chosen] expansion.

[0028] [furthermore, the strategy in which invention corresponding to Claim 3 was beforehand defined in invention corresponding to Claim 2] The node number of fundamental action minded from a starting position is the hierarchical planning method which is the strategy of determining fewest candidates for expansion that may be chosen as a candidate for expansion actually developed.

[0029] [furthermore, the strategy in which invention corresponding to Claim 4 was beforehand defined in invention corresponding to Claim 2 again] It is the hierarchical planning method which is the strategy of determining said candidate for expansion with most node numbers of fundamental action minded from a starting position which may be chosen as a candidate for expansion actually developed.

[0030] On the other hand, in invention corresponding to Claim 1 -4, invention corresponding to Claim 5 is the hierarchical planning method of replacing with and saving a changed part of the state in the node in the state in the node concerned, when the state in the node related by developing action and Gaul changes.

[0031] Moreover, invention corresponding to Claim 6 is set to invention corresponding to Claim 1. When developing action on a procedure network, and the node of Gaul and the expansion knowledge applied to the expansion is lacking, It is the hierarchical planning method of searching for the sequence of the action concerned or fundamental action which attains Gaul by search by trial and error using the information on fundamental action given beforehand.

[0032] Furthermore, invention corresponding to Claim 7 is set to invention corresponding to

Claim 6. At the time of the state ****** same about action or the same node as Gaul as the time of a search start which saves the sequence of fundamental action required in search, the state at the time of the search start, and corresponding action and the information on Gaul It is the hierarchical planning method which develops the sequence of fundamental action required in search, without searching again.

[0033] When [furthermore,] invention corresponding to Claim 8 makes fundamental action sequential execution to the initial state in a starting position again It is the method of generating as a plan the sequence of action which changes a state from an initial state to a goal state. In the hierarchical planning method of obtaining a plan by using expansion knowledge for the sequence which consists of fundamental action the procedure network which makes a node action connected one by one and Gaul, and developing from a starting position It is the hierarchical planning method which expresses action on a procedure network, action in the expansion knowledge over Gaul, and the state of Gaul by two or more stative predicates.

[0034] When developing action on a procedure network, and Gaul, on the other hand in invention corresponding to Claim 8, invention corresponding to Claim 9 as a node for [the] expansion the node of action fundamental from a starting position -- and -- or it is the hierarchical planning method which chooses action connected only through an auxiliary node, and the node of Gaul.

[0035] Moreover, invention corresponding to Claim 10 is the hierarchical planning method which supervises whether the node on a procedure network has lapsed into the inhibiting state defined by the stative predicate in Claim 1 or invention corresponding to 8.

therefore, the time of developing action on a procedure network, and Gaul in the hierarchical planning method of invention corresponding to Claim 1 first -- the node of action fundamental from a starting position as a node for [the] expansion -- and -- or action connected only through an auxiliary node and the node of Gaul are chosen.

[0036] Thus, it will be developed in order about what the state has decided by the nearest Gaul and nearest action being developed in order from a starting position. Therefore, neither Gaul nor action is developed with an argument with the undecided state, or a variable.

[0037] Therefore, since expansion that a state changes can be carried out certainly, making probability high, generation of the plan under which useless action is included can be controlled and the whole plan production efficiency can be raised simultaneously.

[0038] Moreover, in the hierarchical planning method of invention corresponding to Claim 2, when it acts like invention corresponding to Claim 1 and also there are two or more nodes for [which may be chosen] expansion, the candidate for expansion actually developed is determined based on the strategy defined beforehand.

[0039] Therefore, it can plan that it is also at the most suitable expansion strategy.

Furthermore, it sets to the hierarchical planning method of invention corresponding to Claim 3. It acts like invention corresponding to Claim 2, and also the strategy defined beforehand is a strategy of determining the candidate for expansion with few node numbers of fundamental action minded from a starting position which may be chosen as a candidate for expansion actually developed.

[0040] Furthermore, it sets to the hierarchical planning method of invention corresponding to Claim 4 again. It acts like invention corresponding to Claim 2, and also the strategy defined beforehand is a strategy of determining the candidate for expansion with most node numbers of fundamental action minded from a starting position which may be chosen as a candidate for expansion actually developed.

[0041] In the hierarchical planning method of invention corresponding to Claim 5 on the other hand When it acts like invention corresponding to Claim 1 -4 and also the state in the node related by developing action and Gaul changes, a changed part of the state in the node is replaced with and saved in the state in the node concerned.

[0042] Therefore, it is not necessary to save all the states one by one, and effective use of hardware resources can be realized. Moreover, it sets to the hierarchical planning method of invention corresponding to Claim 6. When acting like invention corresponding to Claim 1, and also developing action on a procedure network, and the node of Gaul and the expansion knowledge applied to the expansion is lacking, The information on fundamental action given beforehand is used, and the sequence of the action concerned or fundamental action which attains Gaul is searched for by search by trial and error.

[0043] Therefore, knowledge, such as expansion knowledge, is not perfect, and a planning can be continued also when there is an omission. Furthermore, it sets to the hierarchical planning method of invention corresponding to Claim 7. Act like invention corresponding to Claim 6, and also the sequence of fundamental action required in search, the state at the time of the search start, and corresponding action and the information on Gaul are saved. The sequence of fundamental action required in search is developed at the time of the state ****** same about action or the same node as Gaul as the time of a search start, without searching again.

[0044] Furthermore, in the hierarchical planning method of invention corresponding to Claim 8, action on a procedure network, action in the expansion knowledge over Gaul, and the state of Gaul are expressed by two or more stative predicates again.

[0045] Here, a state technical term describes a state. It becomes possible to express the state about two or more objects which follow, for example, constitute the problem world to one Gaul.

[0046] When by doing in this way describes expansion knowledge, description of expansion knowledge is performed grasping the state about two or more objects. Therefore, the hierarchical planner producer can describe expansion knowledge easily.

[0047] Moreover, it becomes easy to solve by using two or more state technical terms for action or the state representation of Gaul depending on a problem. In the hierarchical planning method of invention corresponding to Claim 9 on the other hand When acting like invention corresponding to Claim 8 and also developing action on a procedure network, and Gaul, as a node for [the] expansion the node of action fundamental from a starting position -- and -- or action connected only through an auxiliary node and the node of Gaul are chosen.
[0048] Moreover, in the hierarchical planning method of invention corresponding to Claim 10, it acts like Claim 1 or invention corresponding to 8, and also it is supervised whether the node on a procedure network has lapsed into the inhibiting state defined by the stative predicate. Therefore, the state where it was forbidden as a constraint is effectively excludable.
[0049]
[Mode for carrying out the invention] The form of operation of this invention is explained hereafter. Drawing 1 is the block diagram showing an example of the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention.
[0050] The knowledge base 10 which this hierarchical planner is realized by the computing system and stored in storage, The knowledge compiler 11 in the processor which compiles the Gaul expansion knowledge 10a and the action expansion knowledge 10b in this knowledge base 10, The available information 12 on the planner engine which was compiled by the knowledge compiler 11 and saved in storage, It is constituted by the planner engine 13 which draws up a plan 15 using the available information 12 on a planner engine if an initial state and the goal state 14 can be given from an input unit.
[0051] Next, operation of the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention constituted as mentioned above is explained. First, the Gaul expansion knowledge and action expansion knowledge for developing the Gaul node and action node on a procedure network which the user described are stored in the knowledge base 10.
[0052] The knowledge stored in this knowledge base 10 is changed into an available form of the planner engine 13 by the knowledge compiler 11. If an initial state and a goal state are inputted into the planner engine 13, a procedure network will be developed applying the knowledge of the knowledge base 10, and a plan 15 will be generated.
[0053] At this time, expansion of the Gaul node with the planner engine 13 etc. is regarded about the Gaul node connected only with the primitive action node and the auxiliary node from the start node on the procedure network. That is, as shown in drawing 2 , only the Gaul node by which the plan is generated is made into an expansion object node, and the Gaul node will decide to perform expansion, if expansion knowledge is applicable.
[0054] Drawing 2 is the figure showing the expansion object node in the procedure network of

the form of this operation. The Gaul node connected with the auxiliary node chisel (this example or branching / joint node 23) to the primitive action node 21 from the start node 20 on the procedure network as shown in this figure, That is, the node makes only the Gaul node by which the plan is generated the Gaul node 24 for expansion.

[0055] Here, since the initial state is given to the start node 20, the state of the primitive action node 21 will have been decided to it, and the planner engine 13 performs Gaul expansion to it based on the definite state which is the action result of the primitive action node 21.

[0056] Therefore, the positive Gaul node expansion will be made, without producing useless action in the process so far at least, when the Gaul node 24 for expansion is developed by the primitive action node. Moreover, the primitive action node developed by doing in this way can also offer a deterministic state by the action. Thereby, the positive Gaul node and action node expansion are realizable.

[0057] Moreover, although there are two or more Gaul nodes for expansion in the procedure network shown in drawing 2 , the expansion strategy in such a case is explained using drawing 3 . Drawing 3 is a figure for explaining the expansion strategy in the procedure network of the form of this operation. <BR [0058]> As shown in this figure, when two or more Gaul nodes 34 for expansion and 35 are on a procedure network, the strategy rule of whether to develop from the Gaul node for expansion in what kind of position is beforehand set as the planner engine 13. As a strategy rule considered here, the breadth-first search expansion strategy which develops the nearest Gaul node 34 for expansion preferentially from the start node 30, for example, and the depth-first search expansion strategy which develops preferentially the furthest Gaul node 35 for expansion from the start node 30 can be considered.

[0059] Therefore, when a breadth-first search strategy rule is set up, the Gaul node 34 for expansion is developed first, and when a depth-first search strategy rule is set up, the Gaul node 35 for expansion is developed first.

[0060] In addition, the expansion strategy rule which can be set up is not limited to a breadth-first search strategy and a depth-first search strategy, and can consider various strategy rules, such as a compounded type expansion strategy which repeats for example, breadth-first search expansion and depth-first search expansion by turns further.

[0061] Next, the state maintenance method in each node is explained. [the conventional planner which does not necessarily develop the Gaul node etc. from a start node] Since primitive action changes a state, if primitive action is performed, before execution, the state where it was not materialized will come to be materialized or that the state where it was materialized conversely stops materializing will arise. Therefore, on a procedure network, when a primitive action node arises, it is necessary to change a formation state about all the nodes after the developed node by expansion of the Gaul node etc., in the conventional planner.

[0062] In the hierarchical planner concerning this embodiment, since expansion progresses one by one from the start node side and the state is decided sequentially from the start node side, it is necessary to register a formation state into no nodes beforehand. So, in the hierarchical planner of this embodiment, only the difference information of the change of state of each primitive action node by the place developed by the primitive action node from the start node is registered.

[0063] Drawing 4 is a figure for explaining how to acquire an enactment state by an expansion object node in this embodiment. As shown in this figure (a), the difference information of the change of state is saved about the primitive action node 41 from the start node 40 to the expansion object node 43.

[0064] Here, in order to investigate whether the Gaul node 43 for expansion can apply expansion knowledge to an expansion object node, it is necessary to get to know the formation state in an expansion object node. It goes back to the node which the state understands, and it understands the formation state of an expansion object node, when difference information is applied from there.

[0065] So, when reference of a formation state is needed in the case of drawing 4 (a), the formation state of the Gaul node 43 for expansion concerned will be drawn from difference information with the node 41 which returns to the start node 40 and the initial state and formation state of the start node 40 understand.

[0066] In this case, there are few nodes which must go back and processing of renewal of a state is shortened sharply. Moreover, since there is little required storage capacity compared with the case where each formation states of all are saved, efficient use of hardware resources is realizable.

[0067] moreover, when expansion of the Gaul node progresses as shown in this figure (b), and the node number from the start node 40 to the Gaul node 44 for expansion increases As shown in this figure, the formation state in the proper primitive action node 43b is saved, and the node number which goes back from the Gaul node 44 for expansion is lessened. In this case, what is necessary is just to return to the primitive action node 43b.

[0068] Next, in the knowledge of existing [the Gaul node], when it cannot develop, the case where the suitable Gaul expansion knowledge is not prepared is explained. In such a case, the formation state in the node in front of the Gaul node is left, and the planner engine 13 applies primitive action which can be performed in the state out of the available information 12 on a planner engine. In addition, to the available information 12 on a planner engine, object knowledge for primitive action to apply, state technical term knowledge, action knowledge, etc. are saved.

[0069] Since a state changes as a result of primitive action application, it goes with the application of primitive action which can be performed in the state further one by one. And the

Gaul node will be replaced with the sequence of the primitive action if the state of the Gaul node is attained.

[0070] Either is chosen and applied when there are two or more primitive actions which can be performed. When search goes wrong, other primitive actions are applied. Failure in search is the case where it does not arrive at Gaul even if it applied fixed primitive action etc., when primitive action which can be performed is lost.

[0071] The situation at this time is shown in drawing 5 . Drawing 5 is the figure showing the situation of the search for developing Gaul in this embodiment.

[0072] Suppose that there is no Gaul expansion knowledge applicable to the available information 12 on the knowledge base 10, i.e., a planner engine, in the state of the primitive action node 52 about the Gaul node 54 for expansion of this figure.

[0073] Therefore, the planner engine 13 searches noting that it performs primitive action which can be performed in the state of the primitive action node 52. As shown in drawing 5 , when the primitive action node sequence 53 which attains the state of the Gaul node 54 for expansion is able to be discovered, this node sequence 53 is replaced with the Gaul node 54 for expansion.

[0074] Moreover, when the replacement to the primitive action node sequence 53 of the Gaul node 54 for expansion takes place by search in this way, Three of the primitive action sequences 53 acquired by the formation state in the Gaul pattern of the Gaul node 54 for expansion and the primitive action node 52 in front of that and search are memorized as internal data of the planner engine 13.

[0075] Replacement of the Gaul node and a primitive action sequence is performed without performing search, when this develops the same Gaul node in the same state in process of future plannings.

[0076] [furthermore, the three above-mentioned information memorized as internal data of a planner engine by search when shown in drawing 5] Since the storage will be lost after a planning is completed, the information about the memorized primitive action sequence is outputted to the file in storage in the same form as the usual expansion knowledge.

[0077] The formation state of the primitive action node 52 Thus, expansion conditions, It enables it to use the new expansion knowledge which makes the Gaul pattern and the primitive action node sequence 53 the contents of expansion for the Gaul pattern of the Gaul node 54 for expansion in the knowledge base 10 also at the time of the next planning.

[0078] Next, the handling of the state technical term of the Gaul pattern is explained. Moreover, in the conventional hierarchical planner, although the Gaul pattern was always expressed by one stative predicate, it enables expression according the Gaul pattern to two or more stative predicates in this embodiment.

[0079] that is, when the object which constitutes the object world of a plan from this

embodiment, for example consists of three objects A, B, and C It enables it to express [essentially] each state of Object A, B, and C in a certain node paying attention to the state existing, for example about all the objects A, B, and C, using [not only the state concerning action of the node but] two or more state technical terms.

[0080] In describing the Gaul expansion knowledge in the case of following, for example, carrying out subgoal expansion of the Gaul node of two or more predicate expression at the contents of expansion, it describes two or more stative predicates as a Gaul pattern of the Gaul node of the contents of expansion.

[0081] Moreover, also when describing the action knowledge for developing the Gaul node of two or more predicate expression in action by a primitive action node similarly, two or more stative predicates are described as a Gaul pattern which is the target of expansion.

[0082] Thus, by describing two or more states, he can describe expansion knowledge, a planner producer recognizing the state of each object, and it becomes easy [creation of expansion knowledge] very [him]. Moreover, it becomes easy to solve a problem to have used two or more stative predicates in this way depending on the problem to deal with.

[0083] in addition, [the planner engine 13] like the case where the Gaul pattern develops the Gaul node expressed by one stative predicate also when developing the Gaul node with the Gaul pattern of two or more stative predicates The expansion knowledge in which expansion conditions are satisfied among the expansion knowledge that the Gaul pattern is the same is applied.

[0084] Next, the case where there is the state where it was forbidden as a constraint on plan creation is explained. In such a case, the inhibiting state is beforehand registered into the file as a list of stative predicates. And if this inhibiting state is read, a new primitive action node is generated by expansion or search of a node and a planner engine changes the formation state in the node on a procedure network It is checked an inhibiting state for the state each time, and when it will be in an inhibiting state, expansion or search of a stop, i.e., just before, is canceled for the operation.

[0085] An interaction will be canceled, if expansion of all the Gaul nodes is completed and a procedure network top serves as only a primitive action node and an auxiliary node through the above processings. The method of detection dissolution of an interaction is as the conventional technology having explained.

[0086] And suitable sequencing is performed when the group of primitive action in the case of becoming an interaction is detected. When sequencing which cancels an interaction cannot be performed, it re-plans.

[0087] Since expansion was made to progress from the start node side according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention as mentioned above, generation of the plan having contained

useless action is controlled. Therefore, the whole plan production efficiency also improves.
[0088] [namely, the thing for which Gaul is developed to a primitive action node as it is also in the state of developing sequentially from Gaul nearest to a start node, and deciding not sequentially from a variable but sequentially from an initial state] Generation of the plan under which useless action is included is controlled, and a plan can be generated with very high certainty to any problems.
[0089] Since the flexibility of expansion is not so high as the conventional method at this time, the creation difficulty of expansion knowledge becomes low and an easy and positive hierarchical planner can be created. Moreover, according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention, efficient plan generation is attained by choosing the expansion strategy of being suitable for a problem. Moreover, when an expansion object node is AND parallel, in breadth-first search and OR parallel, useless expansion can also be controlled by setting up like depth-first search.

[0090] Furthermore, since according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention the update process of an enactment state in each node in accordance with expansion was delayed until it was required, the time of an update process can be shortened sharply. Moreover, since only the node by the side of a start node is updated from an expansion object node also when a formation state needs to be updated, the processing time becomes short.
[0091] Moreover, since the formation state was saved in the proper node position, the time of an update process can be shortened further sharply. Furthermore, [according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention again] Expansion knowledge is not perfect, since it enabled it to perform a planning by search when there was an omission, or also when description of expansion knowledge was difficult, also in inadequate knowledge, a planning becomes possible and, thereby, the scope of a planner spreads in the imperfect knowledge base.
[0092] Moreover, expansion knowledge is not perfect, since the former search result is saved when there is an omission, or also when description of expansion knowledge is difficult, it can develop, without searching the same situation again, and since the same search is not performed twice or more, the production efficiency of a plan can be raised.
[0093] Furthermore, since a search result is not canceled but it was made to save at a file when the above-mentioned search is performed even if the planning was completed, from a next planning, the same search is not repeated but the efficiency of a planning improves.
[0094] Moreover, since two or more predicates were combined when the state of being meaningful was not able to be expressed with one predicate according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of

this invention Also by this case, the Gaul state can be expressed and the descriptive power of knowledge, such as expansion knowledge, can be increased. Therefore, the scope of a hierarchical planner can be extended.

[0095] Furthermore, since the user registered the inhibiting state into the list of inhibiting states according to the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention, others can abolish the necessity of describing knowledge being conscious of an inhibiting state. For this reason, a user's burden of knowledge description decreases. Moreover, since an inhibiting state is detectable immediately, useless search can be excluded. in addition, in the range which is not limited to the form of each above-mentioned implementation, and does not deviate from the summary, many things are boiled and this invention can be deformed

[0096]

[Working example] The example of this invention is explained hereafter. Drawing 6 R> 6 is the block diagram showing an example of the hierarchical planner which applied the hierarchical planning method concerning the example of this invention.

[0097] The knowledge base storage 61 which this hierarchical planner is realized by the computing system, and stores the knowledge base 60, The knowledge compiler 63 in the processor 62 which changes each knowledge in this knowledge base 61 into the rule form which can be processed with a production system 68, The rule storage 65 which stores the production rule 64 compiled by the knowledge compiler 63, It is constituted by the production system 68 in the processor 62 which draws up a plan 67 using a production rule 64 if an initial state and the goal state 66 can be given from an input unit.

[0098] Moreover, the Gaul expansion knowledge 60a which is knowledge for developing the Gaul node, the action expansion knowledge 60b which is knowledge for developing an action node, the object knowledge 60c, and 60d of state technical term knowledge and the action knowledge 60e are saved in the knowledge base 60.

[0099] On the other hand, the planner engine 68a which moves using a production system, and the working memory 68b are established in the production system 68. Thus, the case where a block world problem is dealt with as an example is explained using the hierarchical planner of constituted this example.

[0100] The block world problem is one of the planning problems, and is a problem which transposes the block placed on the table large enough to another state. However, only one can move a block at once. A typical example is shown in drawing 7 .

[0101] Drawing 7 is the figure showing the initial state and goal state of a block world in this example. What is necessary is just to move a block as follows, in order to shift to the goal state 71 from the initial state 70 as shown in this figure.

[0102] 1. Place Block C on a table.

2. Place Block B after Block C.
3. Place Block A after Block B.
[0103] Asking for such a procedure automatically is the purpose of this planning. In this planning, it is necessary to define the object which constitutes the object world first. These contents are made into the object knowledge 60c, and are stored as some knowledge bases 60 in the knowledge base storage 61. In a block world, the object knowledge 60c is described as follows, for example.
[0104]
block Ablock Bblock Ctable Since T, next the physical relationship of a block are expressed, it is necessary to define a stative predicate. It stores in the knowledge base storage 60 by making this into 60d of stative predicate knowledge. The example of description of 60d of stative predicate knowledge is shown below.
[0105]
on(block,block)
ontable(block)
clear(block)
Here, on takes two objects of a block type to an argument, and means that there is a block after that others block. Moreover, ontable and clear take the object of a block type to one argument, and it means that there is anything [no] after a block, respectively, and that a block is on a table. An initial state and a goal state are expressed as follows using these stative predicates.
[0106]
Initial state Goal state on (C, A) on (A, B)
ontable(A) on(B,C)
ontable(B) ontable(C)
clear(B) clear(A)
clear(C)
Furthermore, action is defined as operation of changing a state and it stores in the knowledge base storage 60 as action knowledge 60e.
[0107] put1 in here of "placing after that others block a block", Three kinds of actions of totable of "placing the block after a block on a table" are defined as put2 of "placing after blocking the block on a table" as primitive action. The example of description of action knowledge is shown below.
put1: primitive put1 (block x, block y) precondition clear (x) clear (y) on (x, z) add on (x, y) delete clear (y) on (x, z) put2 : primitive put2 (block x, block y) precondition clear (x) clear (y) ontable (x) add on (x, y) delete clear (y) ontable (x) totable: primitive totable (block x) precondition clear (x) on (x, z) add = clear (z) ontable (x) delete As for the on-(x, z) action put1,

the prerequisite for having the arguments x and y of the object of the block type which is a variable, and performing action is clear (x). clear (y) It is on (x, z). on (x, y) is materialized as a result of action, and it is clear (y). It expresses that on (x, z) stops materializing.

[0108] Next, the Gaul expansion knowledge 60a and the action knowledge 60b for decomposing a target into a more concrete subgoal and more concrete action are described, and it stores in the knowledge base storage 60. The example of description of the Gaul expansion knowledge 60a is shown below.

[0109] the Gaul exhibition ****** -- the 1:goal pattern clear (x)
when on(y,x)
into totable(y)
the Gaul exhibition ****** -- the 2:goal pattern on (x, y)
when on(x,z)
into put1(x,y)
the Gaul exhibition ****** -- the 3:goal pattern on (x, y)
when ontable(x)
into put2(x,y)
here -- the Gaul exhibition ****** -- the Gaul node with a Gaul pattern called clear (x) in the 1 means that it can decompose into totable (y), when on (x, y) is materialized in the node. Here, although one Gaul node is replaced with one primitive action, it is transposed to the subnetwork which generally consists of two or more nodes.

[0110] In the example of this block world, since what is necessary is just to be able to attain on (A, B), on (B, C), ontable (C), and clear (A), the above-mentioned goal state shows that Gaul is developed and a plan can be drawn up, if the above-mentioned Gaul expansion knowledge and action knowledge are used.

[0111] Moreover, the above shows being described with an easy expression rather than inputting each knowledge in the form of a production rule. Next, plan creation operation in the hierarchical planner which was able to give such each knowledge is explained.

[0112] First, the knowledge compiler 63 changes the Gaul expansion knowledge 60a into a production rule 64, referring to the object knowledge 60c, 60d of stative predicate knowledge, and the action knowledge 60e. A production rule 64 is stored in the rule storage 65.

[0113] On the other hand on a production system 68, an initial state and the goal state 66 are first read into a working memory 68b. henceforth -- each node of a procedure network is expressed as one WM element in a working memory 68b (working memory element) -- every -- a procedure network is expressed by a parent node and a child node being registered into WM element. The planner engine 68a performs processing required for this reason.

[0114] For example, the following WM elements are used.
(An element name and int [A node type attribute, an int set parent node, int set child node] A

node ID attribute and int A pattern ID attribute and int)
Notes: int An integral-value type and int set The set type of an integral value is meant.
[0115] Moreover, stative predicate knowledge is also expressed using the following WM elements.
(on and int Fact ID, the string argument 1, string argument 2);
(ontable and int Fact ID, string argument 1);
(clear and int Fact ID, string argument 1);
Notes: string means a string type.
[0116] the above-mentioned Gaul exhibition ****** -- it is as follows when the example which changed the 1 into the production rule 64 is shown notionally.
There is an if Gaul node.
[0117]
The Gaul pattern is clear (x).
Expansion conditions are fulfilled.
then -- WM element showing a new node is generated.
[0118]
A parent node and a child node are set up.
the formation situation of the state in each new node -- setting up .
Here, the planner engine 68a will generate the procedure network which consists of a start node, a Gaul node showing a goal state, and an end node using WM element of a production system 68, if an initial state and the goal state 66 are inputted and this is received.
[0119] Next, the Gaul node is developed with the planner engine 68a using the expansion knowledge changed into the production rule 64. The expansion method is the same as the method explained with the form of implementation of invention.
[0120] First, let only the Gaul node connected only with primitive action, the phantom goal, or an auxiliary node from the start node be the Gaul node for expansion. Moreover, when there are two or more Gaul nodes used as the candidate for expansion, it is set up in initial setting any which develop the Gaul node near a start node preferentially (breadth-first search) the Gaul node far from a start node is developed preferentially (depth-first search), or it chooses they are.
[0121] Furthermore, only the effect by primitive action is registered into the primitive action node, and the state of applying to the target node by deriving a new state reflecting the effect if the state of the last node is decided is searched for.
[0122] On the other hand, when there is no expansion knowledge applicable to the Gaul node for expansion, the planner engine 68a starts the following processings. For example, as shown in drawing 8 , it is a certain primitive action 80, next suppose that there is an expansion [of as / whose Gaul pattern is ontable (A)] object node (Gaul node 81).

[0123] Drawing 8 is a figure for explaining search of action in this example. Here, suppose that there is no expansion knowledge for a state being shown in drawing 8 and attaining ontable (A) as a result of primitive action.
[0124] Then, search is started by the planner engine 68a. A searching function investigates primitive action which can be performed in the state which shows in drawing 8 . Here, it carries out [that three actions shown below can be performed and].
Action An additional list Deletion list ptu1 (A, C) clear (B) on (A, C) clear (C) on(A, B) ptu2 (C, A) on (C, A) clear (A) ontable(C) totable (A) clear (B) ontable (A) on (A, B) -- among these three actions Since totable (A) has ontable (A) in an additional list, if this primitive action is used, it turns out that the Gaul node 81 can be developed. Then, the planner engine 68a chooses this primitive action, and ends search.
[0125] If there is no action which has ontable (A) in an additional list, the planner engine 68a will choose arbitrary actions, will update a state reflecting addition/deletion list, and will ask for action which can be performed again. The fixed number of times repeats this operation. If ontable (A) still is not attained, it backtracks and other actions are tried. Backtracking is repeated, and search is ended, if ontable (A) is not attained even if it searches for all the paths.

[0126] Moreover, this search result is memorized in a working memory 68b. The contents memorized concretely are the sequences of the state at the time of the search start expressed by the list of stative predicates, the Gaul pattern expressed by the stative predicate, and primitive action obtained by search. If search starts, first, the planner engine 68a investigates this storage, and when there is a thing with the same Gaul of the same state, it will develop using the past example, without searching.
[0127] Furthermore, the above-mentioned search result memorized in the working memory 68b is outputted to a file in the form of the Gaul expansion knowledge. Here, as for the state at the time of a search start, expansion conditions and the sequence of primitive action serve as the contents of expansion. Let the Gaul pattern be the Gaul pattern as it is.
[0128] Moreover, the Gaul pattern makes it possible to describe two or more predicates, and also enables the Gaul expansion knowledge to describe two or more predicates according to it. For example, the knowledge which develops the Gaul node which has two predicates, ontable (x) and clear (x), in the Gaul pattern is described as follows.
[0129]
goal pattern ontable(x) clear(x)
when ...
into ...
Here, when (expansion conditions) and into (the contents of expansion) are similarly described to be the above-mentioned examples.

[0130] this knowledge -- a carp -- in the production rule 64 which carried out the pile, only the number of Gaul patterns should increase conditional expression to a condition part corresponding to two or more predicates. Moreover, two or more predicates are bundled with describing the Gaul node which has two predicates, ontable (x) and clear (x), in the Gaul pattern at the contents of expansion by] with [as follows.

[0131]

goal pattern xxxx(x)

when ...

into [ontable(x),clear(x)]

Furthermore, an inhibiting state is expressed as a list of stative predicates, and the check of an inhibiting state is realized by memorizing. If expansion or search takes place and the state of each node is updated, and it compares and investigates whether the state is an inhibiting state and has become an inhibiting state, the expansion or search will be canceled.

[0132] Thus, after expansion is completed and all the nodes turn into a primitive action node or an auxiliary node, the planner engine 63 performs detection and dissolution of an interaction.

[0133] An interaction is detected between parallel primitive actions which are not set in order in search of that with which the prerequisite and effect disagree. When an interaction is detected, primitive action of another side is previously set in order for primitive action by which a prerequisite will be destroyed behind. A plan 67 is outputted if all the interactions are cancelable. When an uncancelable interaction occurs, it re-plans.

[0134]

[Effect of the Invention] As a full account was given above, according to this invention, the generation of a plan containing useless action can be controlled and the hierarchical planning method which raises the whole plan production efficiency simultaneously can be offered.

[0135] Moreover, according to this invention, knowledge is not perfect, and also when there is an omission, the hierarchical planning method which enables it to continue a planning can be offered. Furthermore, according to this invention, the hierarchical planning method which can also cope with the problem expansion knowledge is hard to describe can be offered. On the other hand, according to this invention, the hierarchical planning method which makes it possible to except effectively the state where it was forbidden as a constraint can be offered.

[Brief Description of the Drawings]

[Drawing 1] The block diagram showing an example of the hierarchical planner which applied the hierarchical planning method concerning the form of operation of this invention.

[Drawing 2] The figure showing the expansion object node in the procedure network of the

form of this operation.

[Drawing 3] The figure for explaining the expansion strategy in the procedure network of the form of this operation.

[Drawing 4] The figure for explaining how to acquire an enactment state by an expansion object node in this embodiment.

[Drawing 5] The figure showing the situation of the search for developing Gaul in this embodiment.

[Drawing 6] The block diagram showing an example of the hierarchical planner which applied the hierarchical planning method concerning the example of this invention.

[Drawing 7] The figure showing the initial state and goal state of a block world in this example.

[Drawing 8] The figure for explaining search of action in this example.

[Drawing 9] The figure showing the example which creates a plan candidate using a hierarchical planner.

[Drawing 10] The figure for explaining the check of the interaction in a hierarchical planner.

[Explanations of letters or numerals]

10 -- Knowledge base

10a -- The Gaul expansion knowledge

10b -- Action expansion knowledge

11 -- Knowledge compiler

12 -- Available information on a planner engine

13 -- Planner engine

14 -- An initial state and goal state

15 -- Plan

20, 30, 40, 50 -- Start node

21, 31, 41, 42, 51, 52, 80 -- Primitive action node

22, 32 -- Un-primitive action node

23, 33 -- or branching / joint node

24, 34, 35, 43, 54 -- Gaul node for expansion

25, 36, 44, 55, 81 -- Gaul node

26, 37, 45, 56 -- End node

53 -- Primitive action node sequence

60 -- Knowledge base

61 -- Knowledge base storage

62 -- Processor

63 -- Knowledge compiler

64 -- Production rule

65 -- Rule storage

66 -- An initial state and goal state

67 -- Plan

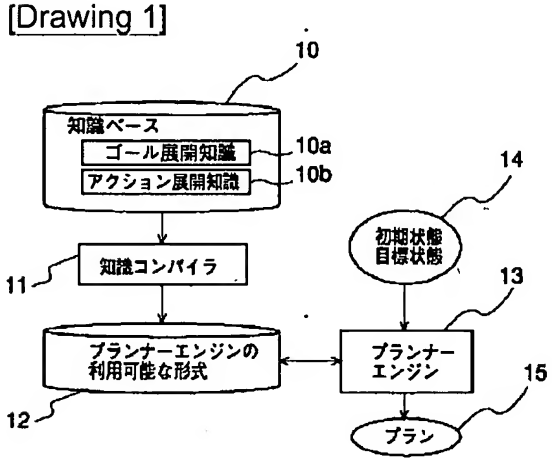
68 -- Production system

68a -- Planner engine

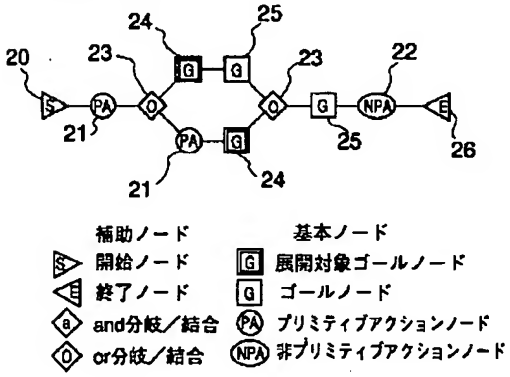
70 -- Initial state

71 -- Goal state

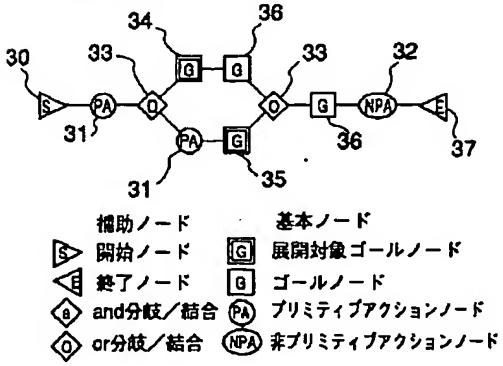
[Drawing 1]



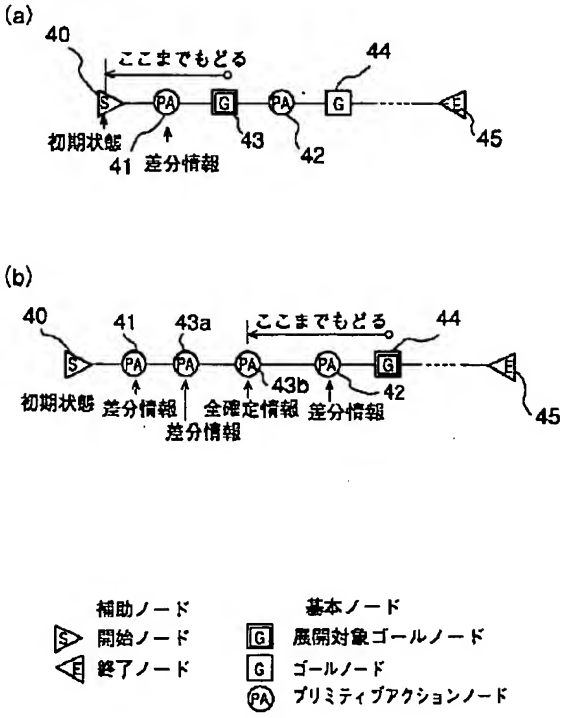
[Drawing 2]



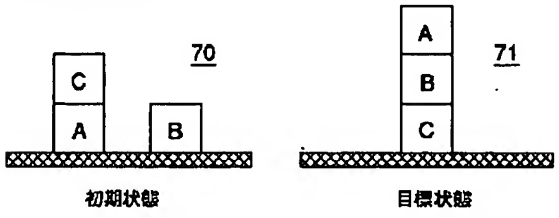
[Drawing 3]



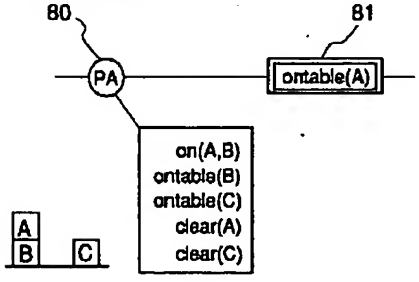
[Drawing 4]



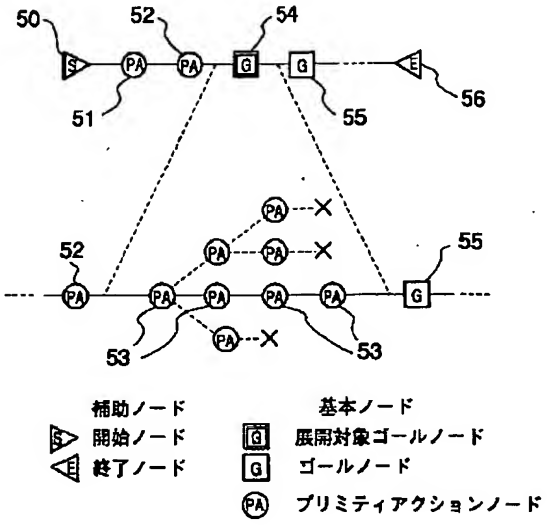
[Drawing 7]



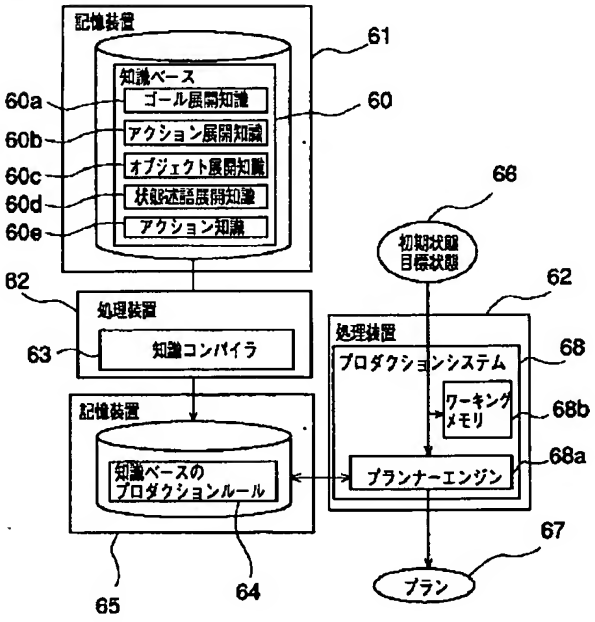
[Drawing 8]



[Drawing 5]

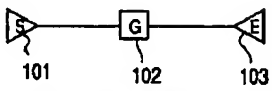


[Drawing 6]

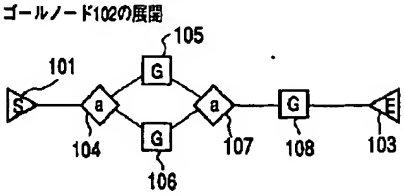


[Drawing 9]

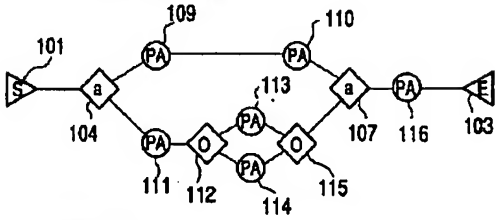
(a) スタート時

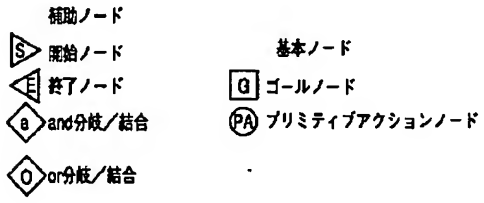


(b) ゴールノード102の展開

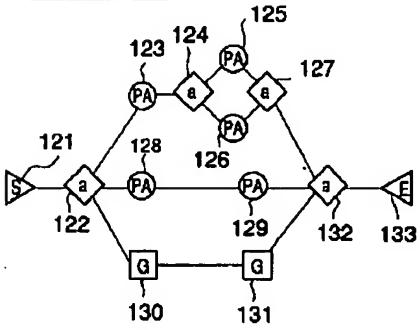


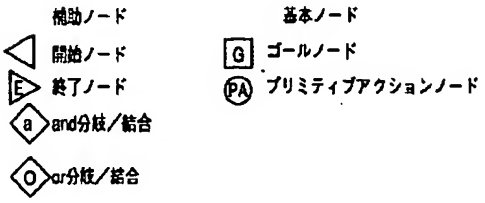
(c) ゴールノード105.106.103の展開





[Drawing 10]





[Translation done.]